SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Juraj Benić

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof.dr.sc. Mladen Crneković

Student:

Juraj Benić

Zagreb, 2017.

Izjavljujem da sam ovaj rad izradio samostalno, koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru prof. dr. sc. Mladenu Crnekoviću što mi je omogućio da napišem ovaj rad, te na stručnim savjetima prilikom izrade rada.

Juraj Benić

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **JURAJ BENIĆ** Mat. br.: 0035188447

Naslov rada na hrvatskom jeziku: **SERVO SUSTAV KAMERE POKRETAN GLAVOM ČOVJEKA**

Naslov rada na engleskom jeziku: **CAMERA SERVO DRIVEN BY HUMAN HEAD**

Opis zadatka:

Upravljanje transportnim sredstvima pomoću video veze otežano je zbog toga što je vidni kut komercijalnih kamera manji od vidnog kuta koji čovjek očekuje (vidno polje je suženo), a time je smanjena količina informacija koju čovjek prima. Da bi se povećalo vidno polje i lakše donosile odluke, kamera bi trebala biti pomična. Kako čovjek vožnju transportnog uređaja najčešće ostvaruje preko upravljačke palice ili tipki, pokretanje kamere preko dodatnih ručnih kontrola moglo bi ga dovesti u situaciju da učini kobnu pogrešku ili ne reagira dovoljno brzo. Zato pomicanje kamere može biti zadano pomicanjem glave čovjeka (gore-dolje, lijevo-desno) što je na neki način i prirodno.

Potrebno je konstruirati servo sustav koji će ostvariti navedeni cilj pomicanja kamere pokretima glave u dvije osi. Uređaj za zadavanje pokreta kamere treba biti ergonomski.

U radu je potrebno:

- odabrati senzore pokreta, prijenos informacija i motore za pokretanje kamere,
- definirati model prepoznavanja orijentacije,
- informaciju o orijentaciji glave bežičnom vezom slati na drugo računalo,
- ostvariti pomicanje kamere,
- izraditi i testirati uređaj te procijeniti investiciju.

Zadatak zadan: 17. studenog 2016.

Rok predaje rada: 19. siječnja 2017.

Predviđeni datum obrane: 25., 26. i 27. siječnja 2017.

Zadatak zadao:

Prof. dr. sc. Mladen Crneković

v. d. predsjednika Povjerenstva:

Prof. dr. sc. Biserka Runje

Sadržaj

Popis slika

Popis tablica

Popis tehničke dokumentacije

Broj crteža	Naziv iz sastavnice
Z−00−01	SKLOP ZA MJERENJE KUTA ZAKRETA GLAVE
Z−01−02	Žiroskop − kontroler
Z−02−01	Žiroskop − tipke
Z−03−00	Žiroskop − napajanje
Z−04−00	Žiroskop − periferija
S−000−001	SKLOP ZA POMICANJE KAMERE
SK−000−002	Sustav za pomicanje kamere
SK−001−000	Nosač donjeg motora
SK−002−000	Nosač gornjeg motora
SK−003−001	Nosač kamere
KU−000−001	Sklop kutije
KU−001−000	Gornja strana kućišta
KU−002−000	Lijeva strana kućišta
KU−003−000	Prednja strana kućišta
KU−004−000	Desna strana kućišta
KU−005−000	Zadnja strana kućišta
KU−006−000	Kutija za baterije
KU−007−000	Poklopac za baterije
K−07−00	Kamera − kontroler
K−08−00	Kamera − napajanje

SAŽETAK

Zbog malog vidnog kuta kamere u odnosu na ljudsko oko došlo se je na ideju da se načini uređaj koji će pomicati kameru na transportnom sredstvu, a bit će upravljan pomacima čovjekove glave (gore-dolje i lijevo-desno). U radu je razrađena ideja te je na kraju i realiziran sam sustav.

Sustav se sastoji od dva dijela. Prvi dio zove se *sklop za mjerenje kuta zakreta glave* i on se sastoji od žiroskopa, kontrolera, bluetooth modula i kacige na koju su smještene sve komponente. Ulaz u sustav je kut zakreta glave, koji se potom obrađuje i šalje bežično na drugi dio sustava. Drugi dio naziva se *sklop za pomicanje kamere*. On se sastoji od bluetooth modula, kontrolera, dva servo motora i kamere. Ulaz u drugi sustav su podatci pristigli preko bluetooth veze između dva sustava te se pomoću njih moduliraju dva signala koji upravljaju servo motorima i pomiču kameru u željenom smjeru.

Ključne riječi: Arduino; ATmega328P; MPU-6050; HM-10 bluetooth; FDM; Python;

ABSTRACT

Due to the small visual angle of a camera in relation to the human eye an idea occurred to make a device that will move the camera on the mobile transport device and it will be controlled by rotating the human head in desired direction. In this paper we elaborated the idea and realization of the system.

The whole system consists of two parts. The first part is called an *assembly for measuring the head rotation angle*. It consists of a gyroscope, controller, Bluetooth module and of a helmet on which all components are placed. The inputs of the system are two angles of head rotation which are then processed and sent wirelessly to the another part of the system. The second part is called an *assembly for moving a camera*. It consists of Bluetooth module, controller, two servomotors and a camera. Inputs to the second system are data received via Bluetooth connection between the two systems, which are then used for modeling two signals for servo motors that are rotating the camera to the desired direction.

Keywords: Arduino; ATmega328P; MPU-6050; HM-10 bluetooth; FDM; Python;

1 UVOD

Razvoj novih tehnologija omogućio nam je da stalno unapređujemo stare stvari te da ih dovodimo do savršenstva. Kroz cijeli taj razvoj prolaze i mobilni transportni uređaji upravljani pomoću video veze. Zbog njihovog stalnog razvoja javila se potreba da se nađe način koji će omogućiti korisniku jednostavno upravljanje kamerom na mobilnom transportnom uređaju. U ovom radu za primjer mobilnog transportnog uređaja uzet će se quadcopter.

1.1 Opis problema

Većina današnjih quadcoptera dolazi s ugrađenom kamerom i omogućava nam upravljanje pomoću video veze. Kamera koja dolazi s quadcopterom obično je ugrađena u samu konstrukciju quadcoptera ili fiksno visi na nosaču ispod konstrukcije samog quadcoptera kao što se to može vidjeti na slici 1. Isto tako postoje i quadcopteri koji mogu pomicati kameru, ali samo upravljanje kamerom vrši se pomoću dodatne upravljačke palice ili s upravljačkom palicom od quadcoptera na kojoj se nalaze dodatne tipke za upravljanje kamerom.

Upravljanje quadcopterom iznimno je teško jer zahtijeva puno vremena da se savlada. Dodamo li tome upravljanje pomoću video veze stvar postaje puno kompliciranija za operatera.

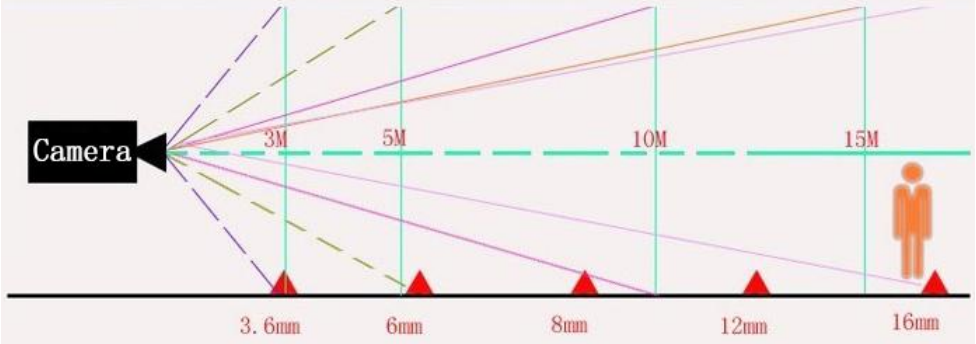
Zbog toga je potrebno osmisliti sustav koji će rotirati kameru na mobilnom transportnom sredstvu radi povećanja vidnog polja operatera, te što jednostavniji način za upravljanje kamerom na mobilnom transportnom sredstvu bez dodatnih upravljačkih palica.

Slika 1: Quadcopter sa kamerom i upravljačkom palicom [7]

1.2 Analiza problema

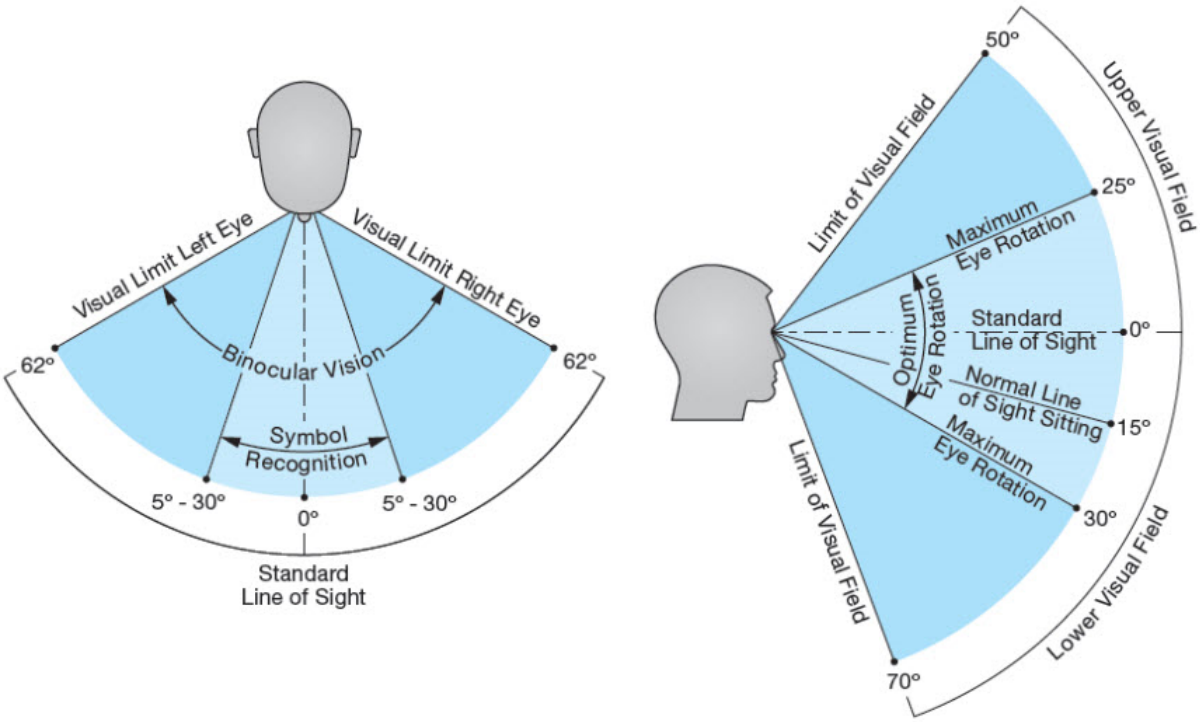
Upravljanje quadcopterom pomoću video veze dodatno je otežano malim vidnim poljem kamere. Vidno polje kamere ovisi o udaljenosti koju želimo vidjeti s kamerom dok udaljenost utječe na odabir leća, a odabir leća utječe na vidno polje kamere. Na slici 2 možemo vidjeti kako se vidni kut kamere (eng. "*Angle*", slika 2) mijenja s odabirom različitih leća (eng. "*Lens*", slika 2).

Lens	4mm	6mm	8mm	12mm	16mm
Angle	69.9°	50°	38.5°	26.2°	19.8°
Distance	0-6M	5-10M	10-20M	20-35M	35-50M



Slika 2: Vidni kut kamere [8]

Vidni kut ljudskog oka može se vidjeti na slici 3. On u horizontalnoj i vertikalnoj ravnini približno iznosi 120° od čega se manje od 60° vidnog kuta odnosi na prepoznavanje simbola (eng. "*Symbol Recognition*", slika 3), dok veći kutevi služe za prepoznavanje oblika i boja. Usporedbom podataka sa slike 2 i slike 3 možemo vidjeti kako je zapravo vidni kut kamere u pola manji od vidnog kuta oka, što nam ne ide u korist kod upravljanja mobilnim transportnim uređajem pomoću video veze.

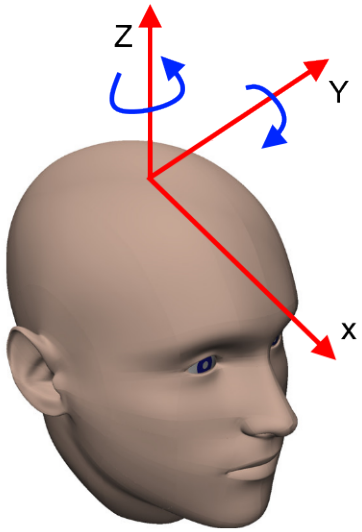


Slika 3: Vidni kut oka [9]

Zbog toga će se u ovom radu razraditi ideja i izraditi sustav koji će pomicati kameru na transportnom sredstvu s ciljem povećanja vidnog polja operatera. Operater će umjesto korištenja dodatne upravljačke palice ili novih tipki na staroj upravljačkoj palici koristiti jednostavne pomake glave (gore-dolje i lijevo-desno) za pomicanje kamere što će mu omogućiti brže reagiranje na promjenu okoline oko mobilnog transportnog uređaja.

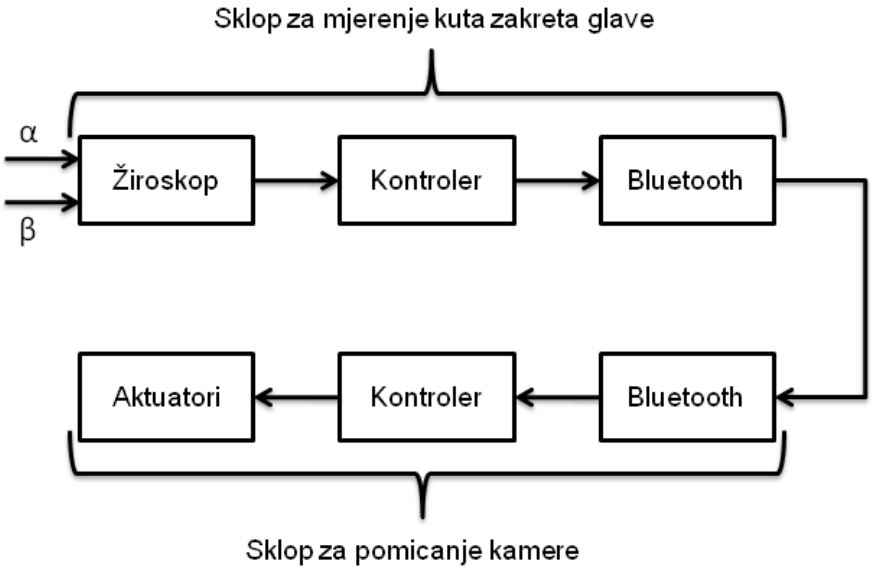
2 RAZRADA IDEJE

Da bi mogli krenuti s razradom ideje najprije moramo postaviti koordinatni sustav na ljudsku glavu. Koristit će se desni koordinatni sustav koji se može vidjeti na slici 4. Sa slike možemo vidjeti da ćemo mjeriti kut zakreta oko Y i Z osi (plave strelice na slici) tj. pomaka glave gore - dolje i lijevo - desno.



Slika 4: Koordinatni sustav glave

Nakon postavljanja koordinatnog sustava moramo konstruirati kacigu. Na istu je potrebno postaviti sve komponente potrebne za mjerenje kuta zakreta glave i za obradu podataka, tj. moramo postaviti žiroskop, bluetooth, izvor napajanja, stabilizatore napona te mikrokontroler. Taj dio zove se *sklop za mjerenje kuta zakreta glave* i može se vidjeti na slici 5. Sklop reagira na dva stupnja slobode gibanja glave koje mjeri žiroskop, a mikrokontroler obrađuje podatke te ih pomoću bluetooth modula prosljeđuje dalje.



Slika 5: Shema sustava za upravljanje kamerom na transportnom sredstvu

Slanjem podataka sa *sklopa za mjerenje kuta zakreta glave* podatci dolaze na *sklop za pomicanje kamere*. Taj sklop se sastoji od bluetooth modula koji prima podatke poslane sa *sklopa za mjerenje kuta zakreta glave* te ih prosljeđuje kontroleru koji ih obrađuje te modulira dva signala za upravljanje aktuatorima. Aktuatore čine dva servo motora koji su konstrukcijski postavljeni tako da omogućuju dva stupnja slobode gibanja konstrukcije.

Konstrukcija na koju će biti postavljena kamera i aktuatori mora biti konstruirana na način da prvi stupanja slobode gibanja rotira kameru u horizontalnoj ravnini, a drugi stupanj slobode gibanja mora omogućavati rotaciju kamere u vertikalnoj ravnini.

3 ODABIR KOMPONENTI

Komponente odabrane za registriranje pomaka glave, obradu podataka i upravljanje kamerom:

1. Za mikrokontroler koji će obrađivati podatke o kutu zakreta glave korisnika odabran je **Atmel-ov 8-bitni ATmega328P mikrokontroler** zbog dovoljnog broja digitalnih ulaza/izlaza te je naravno cjenovno prihvatljiv.

2. Za određivanje kuta zakreta glave koristit će se MEMS senzor **MPU-6050** koji može mjeriti akceleraciju u smjeru osi te kutnu brzinu oko osi.

3. Komunikacija će se ostvarivati putem bluetooth veze te će se stoga koristiti **bluetooth HM-10 moduli**.

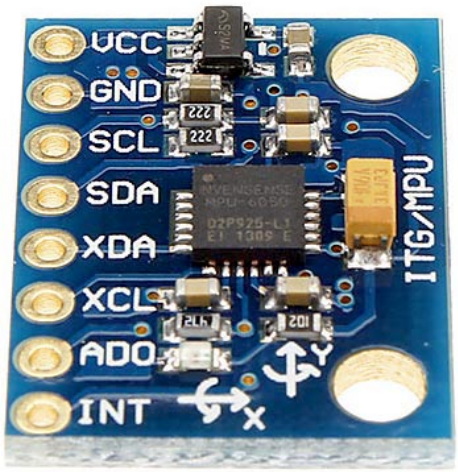
4. Za motore koji će pomicati kameru u vertikalnoj i horizontalnoj ravnini bit će korišteni **Tower Pro SG90** servo motor.

5. Mikrokontroler za upravljanje motorima i primanje podataka bit će isti kao i mikrokontroler za obradu i slanje podataka tj. **Atmel-ov 8-bitni ATmega328P mikrokontroler**.

6. Za kameru će se koristi najobičnija web kamera.

3.1 MPU-6050 žiroskop i akcelerometar

MPU-6050 je 6-osni MEMS senzor koji eliminira potrebu za korištenje posebnog žiroskopa i akcelerometra. Uređaj kombinira 3-osni žiroskop i 3-osni akcelerometar. Podržava spajanje dodatnih senzora preko pomoćnih nadređenih I^2C kanala. Podržava brzinu I^2C komunikacije do 400kHz te ima VLOGIC pin koji definira napon logike MEMS senzora.



Slika 6: MPU-6050 žiroskop i akcelerometar [10]

Tablica 1: Osnovni podatci MPU-6050 senzora

Žiroskop		Akcelerometar	
raspon mjerenja $°/sec$	osjetljivost $LSB/°/sec$	raspon mjerenja g	osjetljivost LSB/g
±250	131	±2	16384
±500	65.5	±4	8192
±1000	32.8	±8	4096
±2000	16.4	±16	2048

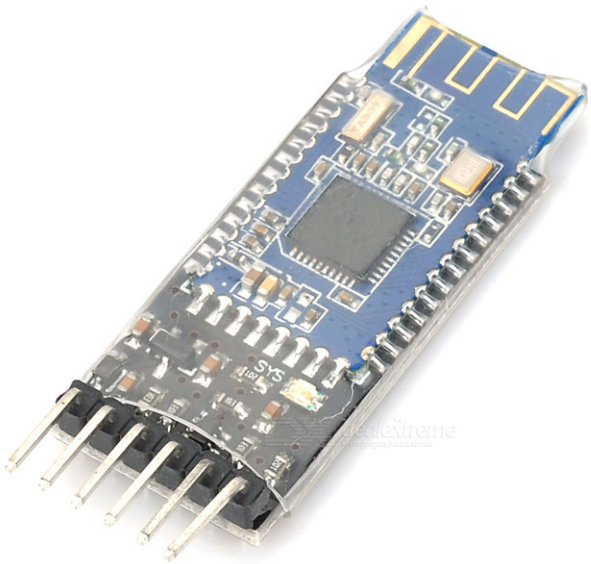
U tablici 2 dan je popis potrebnih registara za rad sa senzorom te njihov kratki opis. Registri će se konfigurirati na način da ćemo u registar *6B* upisati heksadecimalnu vrijednost $0x08$ što znači da će oscilator biti postavljen na $8MHz$, onemogućit će se mjerenje temperature i probuditi uređaj iz "sleep" moda, dok ćemo u registru *6C*, zbog uštede energije, isključiti mjerenje akceleracije. Registar *1B* podesiti ćemo na način da odaberemo najuže mjerno područje što nam daje najveću mjernu preciznost, a iz registara 43-48 iščitavat ćemo vrijednosti kutne brzine.

Tablica 2: Registri potrebni za rad s MPU-6050 senzorom

Registar	Ime	Opis
19	SMPLRT_DIV	Definira brzinu uzorkovanja senzora uz pomoć podatka iz CONFIG registra
1A	CONFIG	Registar služi za postavljanje digitalnog niskopropusnog filtera
1B	GYRO_CONFIG	Služi za podešavanje raspona mjerenja žiroskopa i pokretanje "self-testa"
1C	ACCEL_CONFIG	Služi za podešavanje raspona mjerenja akcelerometra i pokretanje "self-testa"
3B - 3F	ACCEL_OUT	U ove registre pohranjuju se izmjerene vrijednosti akceleracije, brzinom definirane u registru SMPLRT_DIV
43 - 48	GYRO_OUT	U ove registre pohranjuju se izmjerene vrijednosti kutne brzine definiranom u registru SMPLRT_DIV
6B	PWR_MGMT_1	Registar omogućava buđenje uređaja, podešavanje oscilatora te omogućuje resetiranje cijelog senzora
6C	PWR_MGMT_2	Registar omogućuje da isključimo pojedino mjerenje akceleracije u smjeru osi ili kutnu brzinu oko osi
75	WHO_AM_I	Služi za identifikaciju uređaja

3.2 HM-10 bluetooth modul

HM-10 bluetooth moduli mogu se pomoću AT naredba konfigurirati kao *"slave"* i *"master"* te su zbog toga odabrani za korištenje u ovome radu. Komunikacija s kontrolerom odvija se preko serijske komunikacije. Modul posjeduje statusni pin koji registrira da li je modul uparen s drugim ili ne, te isto posjeduje VLOGIC pin koji određuje napon logike. Popis korištenih AT naredbi dan je u tablici 3 zajedno s njihovim opisom.



Slika 7: HM-10 bluetooth modul [14]

Tablica 3: AT naredbe potrebne za rad sa HM-10 bluetooth modulom

Naredba	Opis
AT+BAUD[parametar]	Postavlja brzinu serijske komunikacije
AT+NAME[parametar]	Postavlja ime uređaja, maksimalno 12 znakova
AT+ROLE[parametar]	Postavlja uređaj u *"master"* ili *"slave"* mod
AT+DISC?	Modul počinje tražiti uređaje u svojoj blizini.
AT+CON[parametar]	Uređaj se spaja s drugim tako da se za parametar upiše *MAC* adresa drugog uređaja

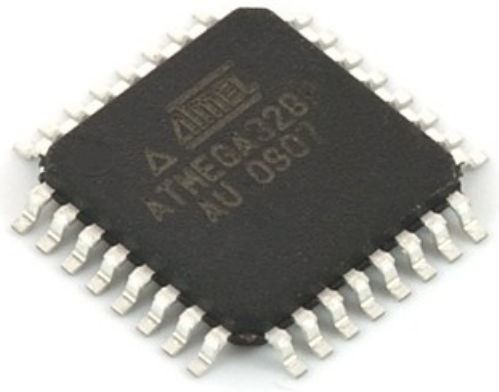
Algoritam 1 služi za konfiguraciju bluetooth modula uz pomoć kontrolerske platforme Arduin-o UNO. Arduin-o UNO čita AT naredbe zadane na računalu te ih prosljeđuje bluetooth modulu, a nama vraća odgovor koji nam govori da li je naredba prihvaćena ili ne.

```
1  #include <SoftwareSerial.h>
2  #define TX 8
3  #define RX 9
4
5  SoftwareSerial BTserial(RX, TX);
6  char c = ' ';
7
8  void setup()
9  {
10     Serial.begin(9600);
11     BTserial.begin(9600);
12 }
13
14 void loop()
15 {
16     if (BTserial.available())
17     {
18         c = BTserial.read();
19         Serial.write(c);
20     }
21     if (Serial.available())
22     {
23         c =  Serial.read();
24         BTserial.write(c);
25     }
26 }
```

Algoritam 1: Algoritam za konfiguraciju bluetooth modula

3.3 ATmega328P mikrokontroler

ATmega328P mikrokontroler ima 14 digitalnih ulaza/izlaza te 6 analognih ulaza. Kontroler podržava bootloader koji služi za programiranje kontrolera pomoću USB-a i zamjenjuje vanjski programator. Programira se pomoću Arduionovog besplatnog i open-source programa koji dolazi sa nekolicinom jednostavnih primjera za početnike. Jezik za programiranje baziran je na C/C++ jeziku.



Slika 8: ATmega328P [15]

Tablica 4: ATmega328P - tehničke specifikacije

Mikrokontroler	ATmega328P
Radni napon	5V
Digitalni U/I pinovi	14 (6 ih pruža mogućnost pulsno širinske modulacije)
PWM digitalni pinovi	6
Analogni pinovi	6
DC struja po pinu	50mA
FLASH memorija	32KB (0,5KB odlazi na bootloader)

3.4 Tower Pro SG90 servo motor

Tower Pro SG90 servo motor često je korišten u daljinski upravljanim mobilnim transportnim sredstvima kao što su avioni, helikopteri i autići zbog svoje iznimno male težine i malih dimenzija. Njegove karakteristike dane su u tablici 5.

Slika 9: Tower Pro SG90 servo motor [11]

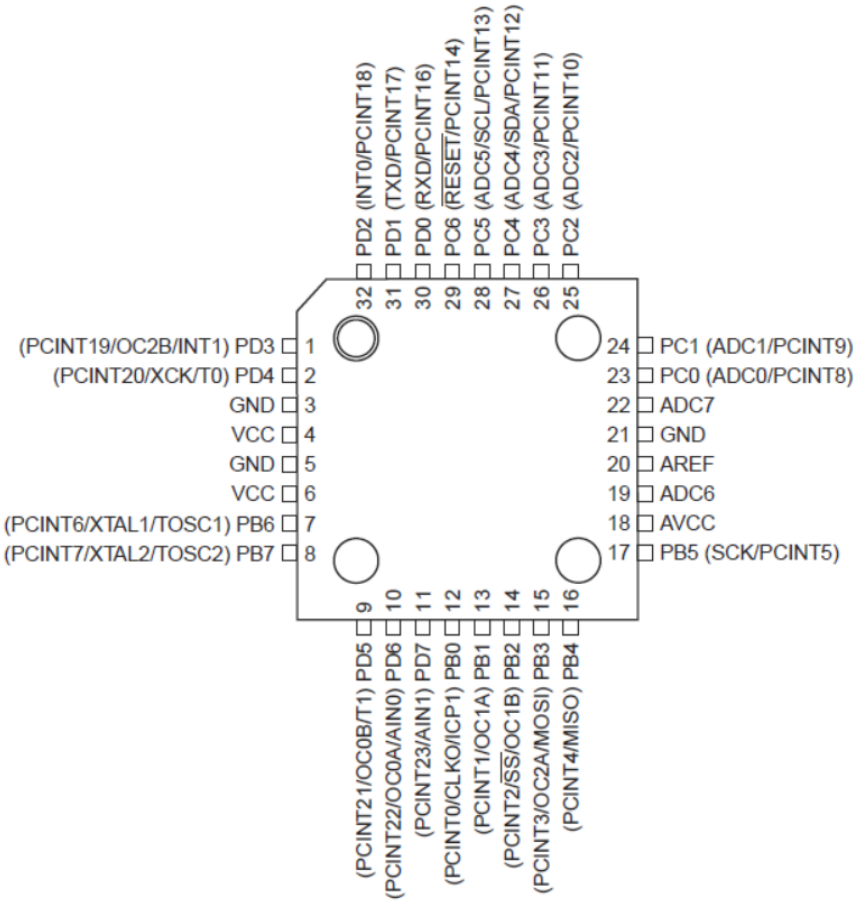
Tablica 5: Tower Pro SG90 - tehničke specifikacije

Radni napon	3,0 - 7,2V
Brzina	$0,12\ sec/60deg$ za $4,8\ V$
Težina	$9\ g$
Moment	$1,2\ kg/cm$ za $4,8\ V$
Dimenzije	$22x11,5x27\ mm$

4 IZRADA SUSTAVA

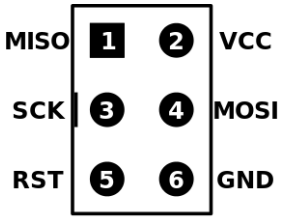
4.1 Konstrukcija kontrolera

Na slici 10 možemo vidjeti značenje svakog pina kontrolera te kako ga moramo spojiti. Pinove 3, 5 i 21 spajamo na GND, a pinove 4, 6, 18 i 20 spajamo na $5V$.



Slika 10: Prikaz pinova za ATmega328P [3]

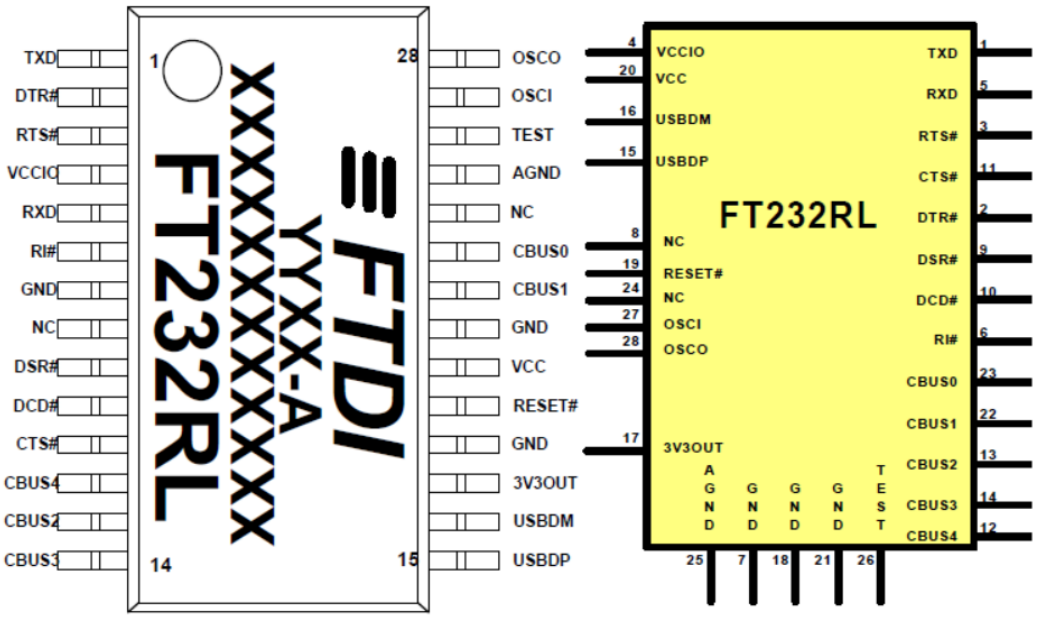
ISP (eng: in-system programming) konektor se spaja na pinove 15, 16, 17 i 29 i to redoslijedom prikazanim na slici 11. Oscilator se spaja na pinove 7 i 8 te se na njih spajaju još dva kondenzatora od $22pF$ čiji se drugi krajevi spajaju na GND.



Slika 11: ISP konektor [16]

Serijska komunikacija između ATmega328P i računala bit će ostvarena pomoću USB-a te nam je stoga potreba čip za komunikaciju. Uzet ćemo FT232RL čip čije se podnožje vidi na slici 12. Pinovi 30 (RXD) i 31 (TXD) s mikrokontrolera spajaju se na pinove 1 (TXD) i 5 (RXD) dok se pin 29 spaja preko kondenzatora od $100nF$ na pin 2 (DTR).

Pinovi 7, 18, 21, 25 i 26 FT232RL čipa spajaju se na GND, a pinovi 4 i 20 na $5V$. Pin 17 spaja se preko kondenzatora od $10nF$ na GND, a pinovi 15 i 16 USB D- i D+. Na pinove 22 i 23 spajaju se dvije LED diode koje svijetle kada se odvija komunikacija.



Slika 12: Raspored pinova za FT232RL čip [4]

Kod miktokontrolera korištenog za mjerenje kuta zakreta glave na pinove 23, 24 i 25 spojit ćemo tipke za pokretanje, zaustavljanje i ponovno pokretanje uređaja dok se senzor spaja na pinove 27 i 28, a bluetooth na pinove 14 (statusni bit), 12 i 13 koji će se programski modulirati kao serijska komunikacija. LED dioda se spaja na pin 10, a zujalica na pin 11.

Kod mikrokontrolera korištenog za upravljanje kamerom signali za upravljanje servo motorom spajaju se na pinove 13 i 14 dok će se bluetooth spajati na pinove 10 (statusni bit), 9 i 11, a signalna LED dioda na pin 1.

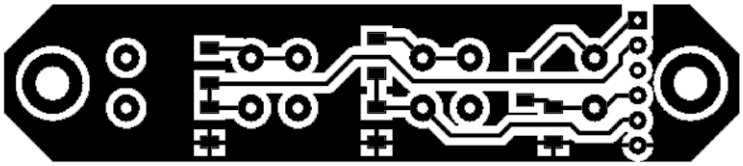
Cijeli sklop je potrebno napajati s $5V$ i $3,3V$ stoga će se koristiti dva stabilizatora napona:

1. LM7805 - $5V$ stabilizator napona
2. LM1117MP3.3V - $3,3V$ stabilizator napona

4.2 Izrada tiskanih pločica

Tiskane pločice izradit će se foto postupkom na fotopozitivnim tiskanim pločicama. Redoslijed kojim se izrađuju pločice je sljedeći:

1. Potrebno je na paus papir ili prozirnu foliju isprintati layout pločice koju želimo izraditi.



Slika 13: Izgled layouta za izradu tiskane pločice

2. Nakon što smo isprintali layout stavljamo papir na staklo tako da je strana papira na koju je printano okrenuta prema pločici, te pločicu osvijetlimo sa ultraljubičastim svjetlom, u ovom slučaju bilo je potrebno osvjetljavati $13min$.

Slika 14: Osvjetljavanje fotopozitivne pločice

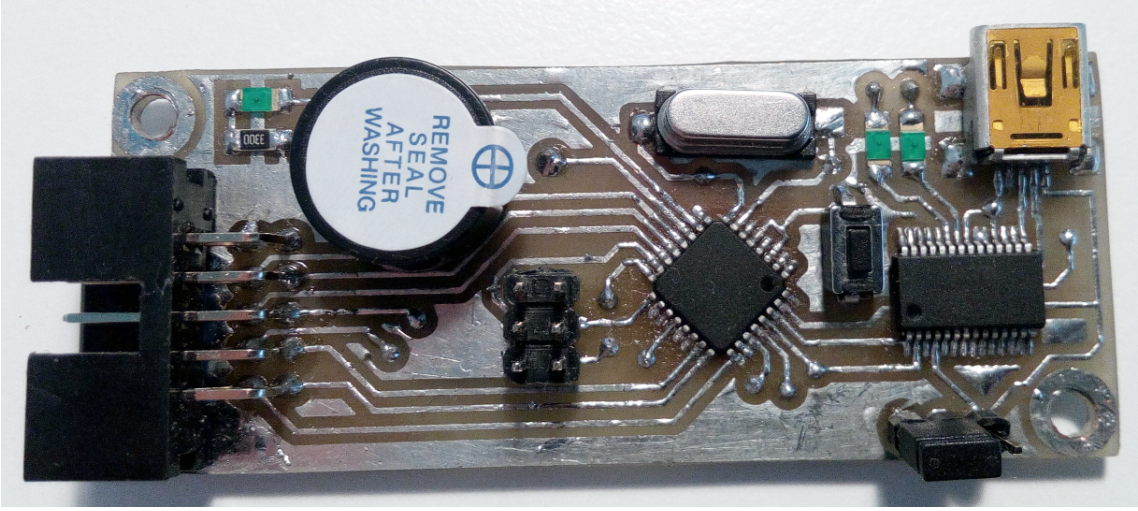
3. Zatim slijedi razvijanje pločice što se radi pomoću natrijevog hidroksida ($NaOH$) otopljenog u vodi. Pločica se razvija u plastičnoj posudici tako dugo sve dok se vodovi ne vide jasno na njoj.

4. Nakon uspješnog razvijanja slijedi jetkanje pločice koje se radi s ferokloridom otopljenim u vodi ili vodikovim peroksidom pomiješanim sa solnom kiselinom. Jetkanje se isto vrši u plastičnoj posudici. Pločica je gotova tek kada sav bakar nestane sa područja na kojima se ne vide vodovi.

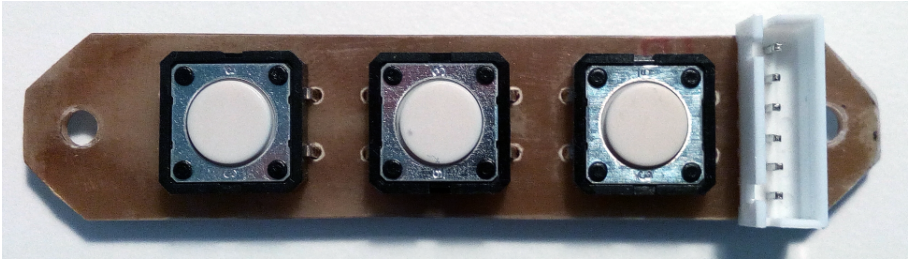
5. Nakon jetkanja bakra moramo pločicu očistiti sa acetonom ili nitro razrijeđivačem, zatim pokositriti da bakar ne bi oksidirao tijekom vremena.

6. Zadnje što moramo napraviti su provrti i lemljenje komponenti na pločicu.

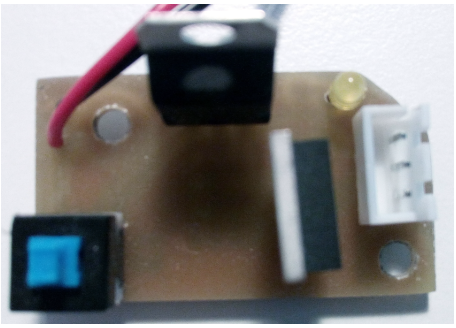
Sa slika 15, 16, 17 i 18 mogu se vidjeti sve izrađene pločice potrebne za određivanje kuta zakreta glave.



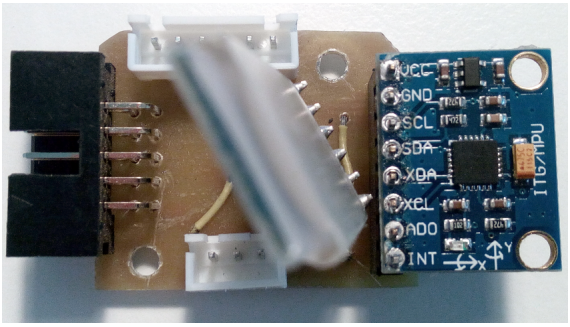
Slika 15: Žiroskop - kontroler



Slika 16: Žiroskop - tipke



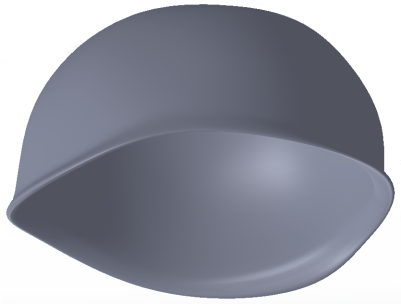
Slika 17: Žiroskop - napajanje



Slika 18: Žiroskop - periferija

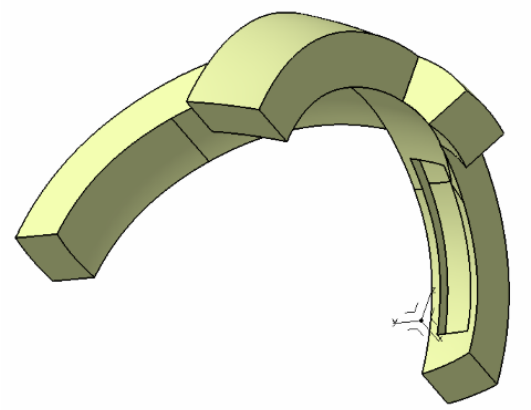
4.3 Konstrukcija kacige

Kaciga je konstruirana po modelu kacige iz Drugog svjetskog rata čiji je model preuzet s GrabCAD-a [17], a odabrana je iz razloga što njen model vjerno odgovara ljudskoj glavi dok su modeli novih kaciga nacrtani bez spužve u kacigi te time ne odgovaraju vjerno ljudskoj glavi. Od modela kacige preuzeta je njena unutarnja površina koja je ujedno i referentna površina konstruiranog modela kacige. Cijeli model napravljen je u Catia-i.



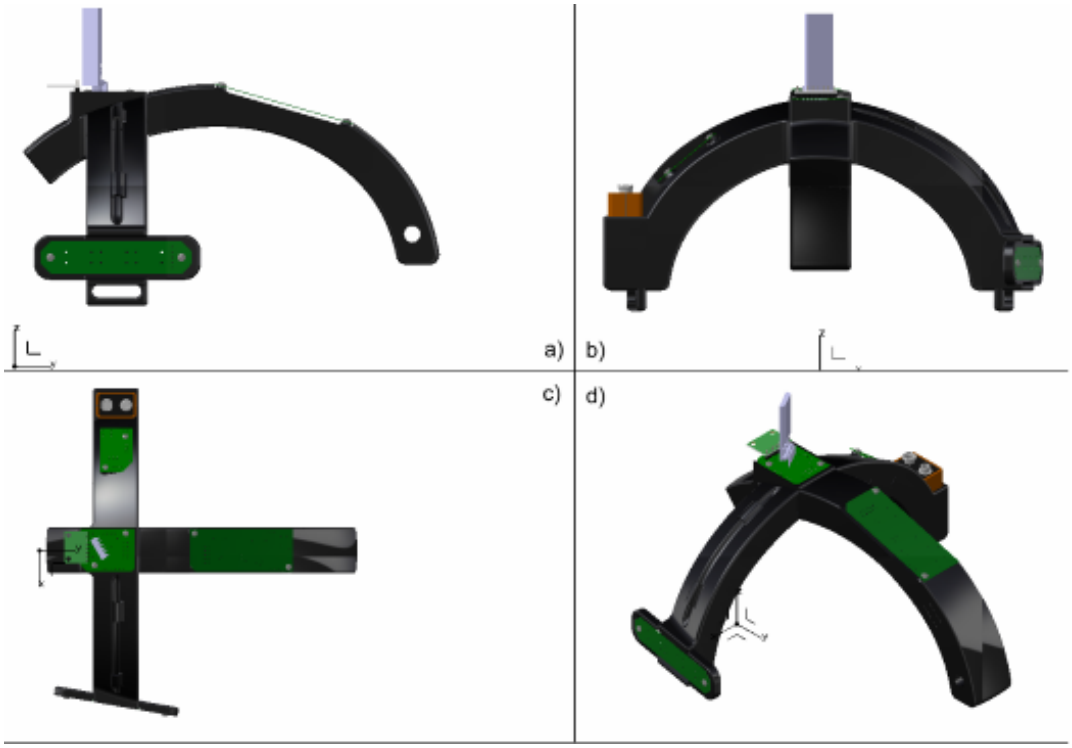
Slika 19: Model referentne kacige [17]

Modela kacige većinom je konstruiran pomoću površina što se može vidjeti na slici 20. Konstruiralo se je pomoću površina jer model kacige ima vrlo kompleksu geometriju koju je bilo jednostavnije modelirati pomoću površina umjesto da se je modeliralo s tijelima.



Slika 20: Model kacige napravljen pomoću površina

Nakon što smo izmodelirali kacigu s površinama i dobili zatvorenu geometriju, koja je postala tijelo, na kacigi su se još modelirala mjesta za postavljanje tiskanih pločica, kanal za smještaj žica, utor za bateriju, nosač za tipke te dva mjesta za pričvršćenje remena. Cijela kaciga je podijeljena na šest dijelova kako bi stala u radni prostor 3D printera. Dijelovi će biti međusobno spojeni lijepljenim spojem, dok je jedan spoj modeliran kao snap-fit spoj. Izgled kacige vidi se na slici 21.



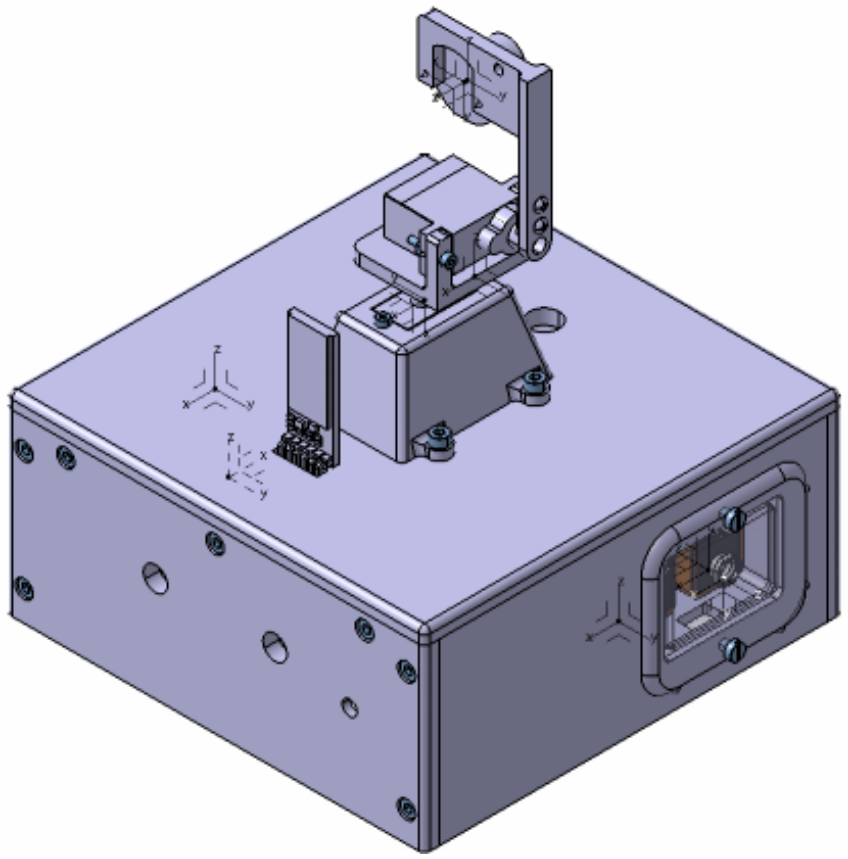
Slika 21: Konstruirana kaciga: a) nacrt, b) bokocrt, c) tlocrt i d) izometrija

4.4 Konstrukcija sklopa za pomicanje kamere

Inspiracija za izradu konstrukcije za pomicanje kamera pronađena je u jednoj od komercijalnih konstrukcija koja se može vidjeti na slici 22. Donji motor će rotirati kameru oko Z osi, dok će gornji motor rotirati kameru oko Y osi. Cijela konstrukcija bit će konstruirana isto kao i kaciga pomoću CAD programa Catia.

Slika 22: Komercijalna konstrukcije za pomicanje kamere [12]

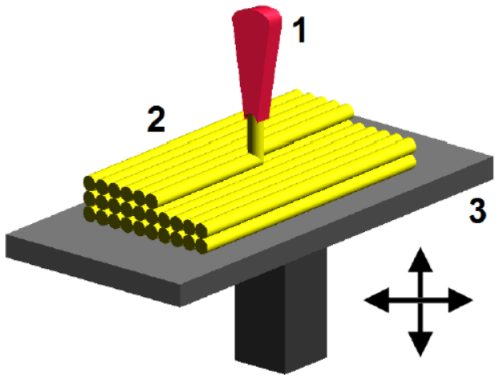
Na slici 23 možemo vidjeti kako izgleda sklop za pomicanje kamere. Cijeli sklop se sastoji od 13 konstruiranih dijelova, 60 standardnih dijelova, kamere, dva servo motora, bluetooth modula te dvije baterije. Dijelovi su međusobno spojeni vijčanim spojevima.



Slika 23: Konstruirani sklop za pomicanje kamere

4.5 Izrada kacige i sklopa za pomicanje kamere

Kaciga i konstrukcija za pomicanje kamere izrađeni su pomoću 3D printera koji radi na FDM principu. FDM metodu (eng. "fused deposition modeling", hrv. "modeliranje topljenim depozitima") razvili su Scott i Lisa Crump krajem 80-tih godina, a komercijalizirana je 90-tih godina. Proces započinje snimanjem CAD modela u STL format koji se zatim orijentira u radnom prostoru printera. Orijentirani model se zatim siječe na slojeve te se stvara G-kod za izradu modela. Model je proizveden ekstrudiranjem malih spljoštenih niti rastaljenog materijala koji formiraju slojeve, a materijal se skrućuje odmah nakon izlaska iz mlaznice. Princip rada vidi se na slici 24.



Slika 24: FDM metoda: 1 - sapnica za izbacivanje rastaljenog materijala, 2 - depozit materijala (modelirani dio) i 3 - pomični stol [5]

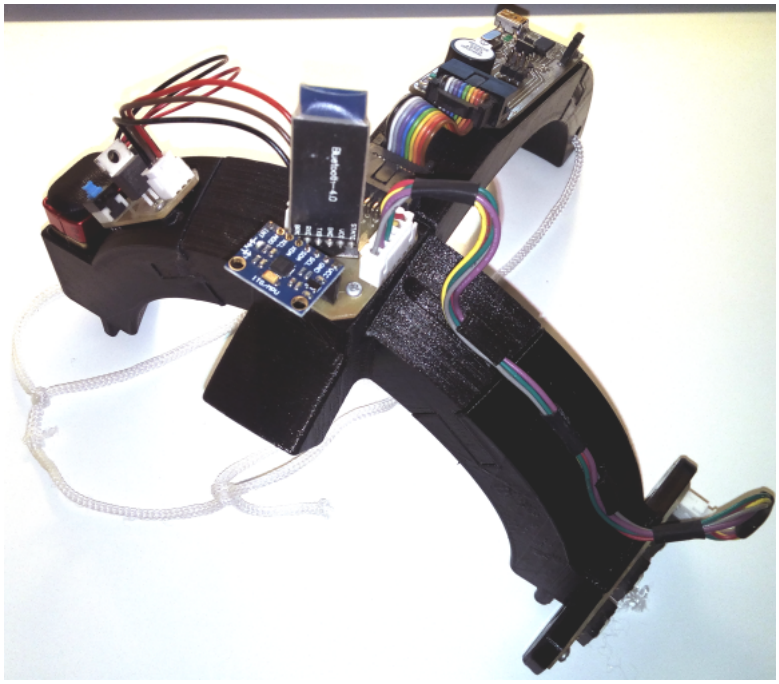
Dijelovi kacige i konstrukcije za pomicanje kamere printani su na vlastitom printeru koji je napravljenom po uzoru na RepRapPro Mendel Tricolour [13]. Slika 25 prikazuje kako izgleda isprintani dio kacige koji se još uvijek nalazi na grijačoj ploči printera.

Slika 25: Printanje centralnog dijela kacige

Na slici 26 možemo vidjeti isprintane dijelove kacige koje je potrebno međusobno spojiti dok na slici 27 možemo vidjeti kako izgledaju međusobno spojeni dijelovi sa

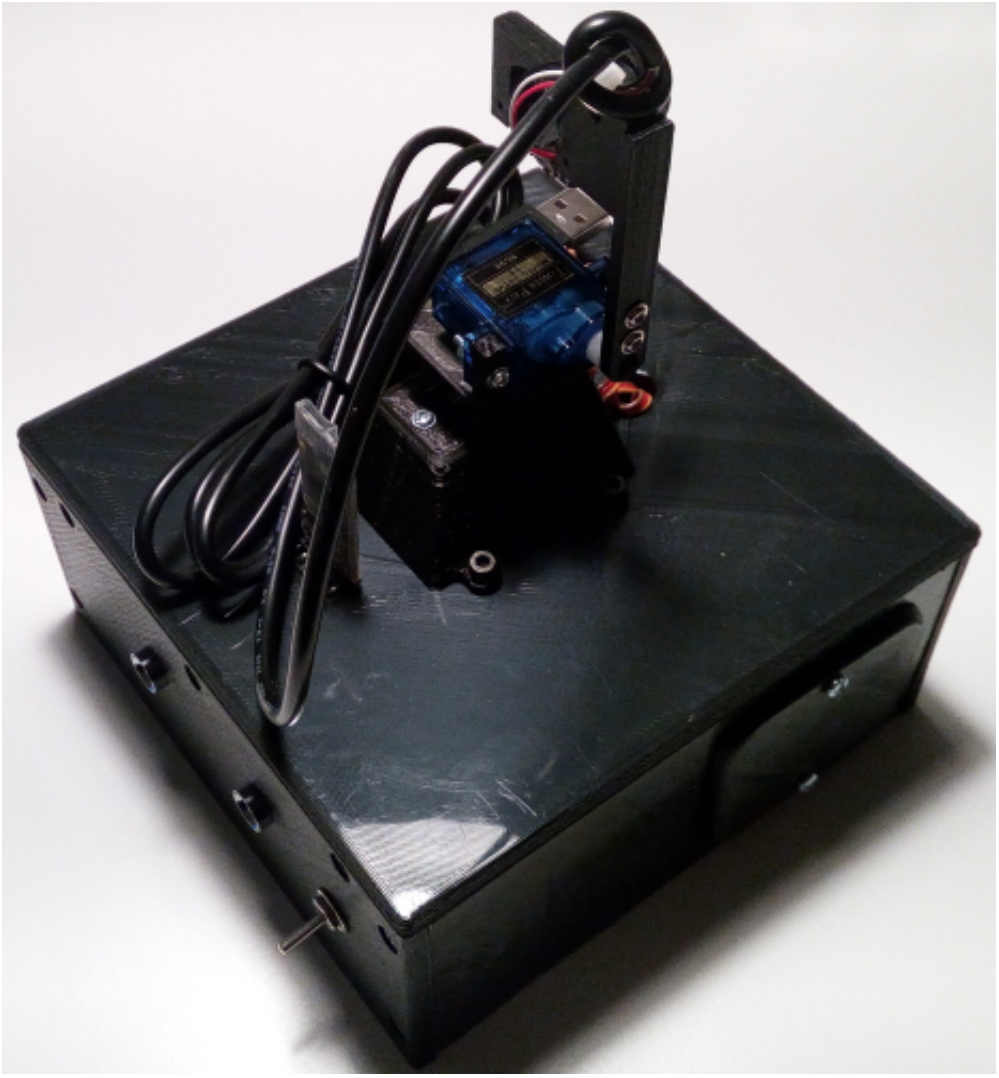
slike 26 i tiskane pločice koje su pričvršćene na kacigu pomoću vijaka.

Slika 26: Isprintani dijelovi kacige



Slika 27: Sklop kacige

Sa slike 28 možemo vidjeti kako izgleda izrađeni sklop za pomicanje kamere.



Slika 28: Sklop za pomicanje kamere

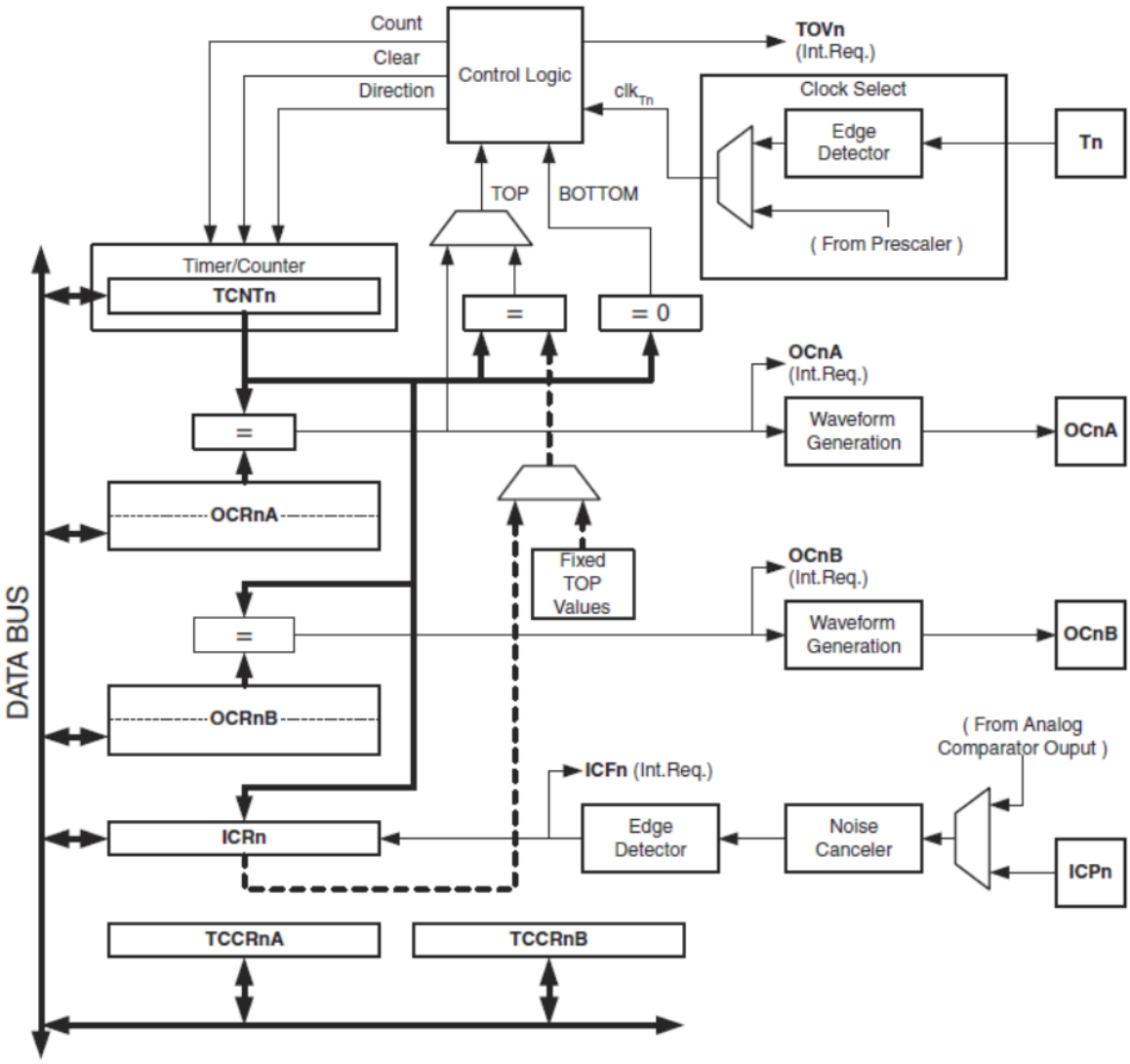
5 PROGRAMIRANJE TAJMERSKIH REGISTARA

ATmega328P mikrokontroler ima dva 8-bitna i jedan 16-bitni tajmer/brojač. Oni omogućavaju precizno izvršavanje programa, pulsno-širinsku modulaciju signala (PWM) te mjerenje vremena signala. Tajmer 0 koristi se kod funkcija *delay()*, *millis()* i *micros()* u softverskom paketu Arduino, a tajmer 2 se koristi za funkciju *tone()*. 16-bitni tajmer 1 se ne koristi te je on slobodan za programiranje u vlastitim funkcijama. Tajmeri se programiraju pomoću tajmerskih registara.

5.1 Opis registara

Na slici 29 možemo vidjeti blok dijagram tajmera/brojača. Malo slovo n u nazivu registra na slici 29 kod programiranja tajmera treba zamijeniti za brojem tajmera koji se koristi. Tajmer/brojač 1 se sastoji od sljedećih registara:

- TCNT1 registar (slika 29) u koji se sprema trenutna vrijednost brojača/tajmera.
- OCR1A/B registara (slika 29) u koje se upisuje izlazna vrijednost za uspoređivanje sa TCNT1 registrom. Vrijednost OCR1A/B registra se stalno uspoređuje sa registrom TCNT1 te se rezultat uspoređivanja može upotrijebiti za generiranje PWM signala.
- ICR1 registar (slika 29) koji služi za detektiranje vanjskog signala te može biti podešen na način da detektira rastući ili padajući brid signala.
- u TIFR1 registru se mogu vidjeti svi zahtjevi za prekide, a u TIMSK1 registru se prekidi uključuju postavljanjem pojedinih bitova u nulu ili jedinicu. Registri TIFR1 i TIMSK1 nisu vidljivi na slici 29.
- TCCR1A/B/C registri su tajmerski kontrolni registri i služe za podešavanje tajmera.



Slika 29: Blok dijagram 16-bitnog tajmera/brojača 1 [3]

Ovisno o načinu rada tajmer se može ponovo pokrenuti, uvećati ili smanjiti za jedan u svakom taktu mikrokontrolera. Takt može biti generiran izvana pomoću vanjskog oscilatora ili može biti korišten unutarnji oscilator.

5.2 Načini rada tajmera

Načini rada ili ponašanje tajmera i izlaznih OC1A/B pinova za usporedbu definira se pomoću moda za generiranje signala (WGM) i moda za uspoređivanje izlaza (COM) postavljanjem WGM i COM bitova u nule i jedinice. COM bitovi ne utječu na mod brojanja tajmera dok WGM bitovi utječu. Popis svih načina rada tajmera i

kombinacija WGM bitova za dane načine rada mogu se vidjeti u tablici 6. Pomoću COM bitova kontrolira se PWM signal, tj. da li će izlazni signal biti invertiran ili ne, te da li će uopće biti generiran. U ovom radu neće se obraditi svi načini rada već samo oni korišteni u radu.

Tablica 6: Načini rada tajmera [3]

Mod	WGM13	WGM12	WGM11	WGM10	Način rada tajmera	Vrijednost do koje broji
0	0	0	0	0	normalni način rada	$0xFFFF$
1	0	0	0	1	PWM, korekcija faze, 8-bitni	$0x00FF$
2	0	0	1	0	PWM, korekcija faze, 9-bitni	$0x01FF$
3	0	0	1	1	PWM, korekcija faze, 10-bitni	$0x03FF$
4	0	1	0	0	CTC	OCR1A
5	0	1	0	1	brzi PWM, 8-bitni	$0x00FF$
6	0	1	1	0	brzi PWM, 9-bitni	$0x01FF$
7	0	1	1	1	brzi PWM, 10-bitni	$0x03FF$
8	1	0	0	0	PWM, korekcija faze i frekvencije	ICR1
9	1	0	0	1	PWM, korekcija faze i frekvencije	OCR1A
10	1	0	1	0	PWM, korekcija faze	ICR1
11	1	0	1	1	PWM, korekcija faze	OCR1A
12	1	1	0	0	CTC	ICR1

5.2.1 Normalni način rada

Normalni način rada ujedno je i najjednostavniji način rada tajmera. Da bismo ga odabrali moramo postaviti WGM bitove u nulu kao što se može vidjeti u tablici 6. U ovom modu tajmer uvijek uvećava svoju vrijednost za jedan te se vrijednost brojača nikad ne briše. Tajmer jednostavno prijeđe preko svoje maksimalne vrijednosti (maksimalna vrijednost 16-bitnog broja iznosi $0xFFFF$) te zatim kreće od svoje minimalne vrijednosti $0x0000$. TOV1 zastavica u normalnom načinu rada služi kao 17-ti bit koji poziva prekidnu rutinu. Zastavica se postavlja u istom trenutku kada i tajmer na svoju donju vrijednost, a briše čim uđemo u prekidnu rutinu. U TCNT1 registar se u normalnom načinu rada može u svakom trenutku upisati nova vrijednost te time podesiti tajmer da broji željeno vrijeme.

U tablici 7 možemo vidjeti koje nam sve funkcionalnosti nudi postavljanje COM bitova u pojedino stanje.

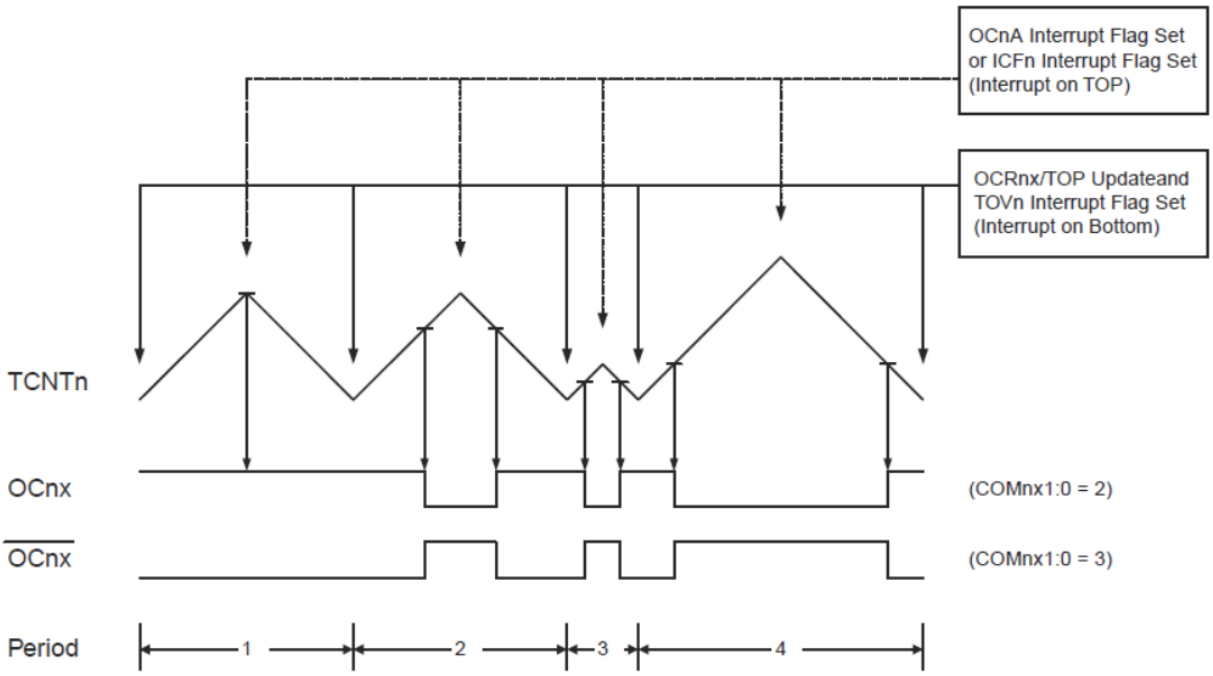
Tablica 7: Funkcionalnost COM bitova [3]

COM1A1/COM1B1	COM1A0/COM1B0	Opis funkcije
0	0	normalni način rada, OC1A/OC1B odspojeni
0	1	Prebacivanje izlaza OC1A/OC1B iz visokog u nisko stanje i obratno
1	0	Postavljanje izlaza OC1A/OC1B u nisko stanje
1	1	Postavljanje izlaza OC1A/OC1B u visoko stanje

5.2.2 PWM mod sa korekcijom faze i frekvencije

Mikrokontroler nudi dva PWM moda s korekcijom faze i frekvencije koji se razlikuju po tome da li se vrijednost do koje broje nalaze u ICR1 ili OCR1A registru kao što se to može vidjeti u tablici 6 pod modovima 8 i 9. U ovim modovima brojač broji od donje ($0x0000$) do maksimalne vrijednosti te od maksimalne do donje vrijednosti. U ne invertirajućem modu izlaz OC1x postavlja se u nisko kada dođe do

usporedbe između TCNT1 i OCR1x registra pri brojanju do maksimalne vrijednosti, a postavlja se nazad u visoko stanje kod brojanja u nazad. U invertirajućem modu postavljanje izlaza je obrnuto. Izgled ne invertirajućeg i invertirajućeg moda može se vidjeti na slici 30 za različite vrijednosti OCR1x i TCNT1 registra, a potrebnu kombinaciju COM bitova za odabir odgovarajućeg izlaza možemo vidjeti u tablici 8.



Slika 30: Dijagram postavljanja izlaza u nisko i visoko stanje za PWM mod sa korekcijom faze i frekvencije [3]

Tablica 8: Funkcionalnost COM bitova za PWM mod sa korekcijom faze i frekvencije [3]

COM1A1/COM1B1	COM1A0/COM1B0	Opis funkcije
0	0	normalni način rada, OC1A/OC1B odspojeni
0	1	Za modove 9 i 11 prebacivanje izlaza OC1A iz visokog u nisko stanje dok je OCR1B odspojen, za sve ostale modove normalan način rada
1	0	Postavljanje izlaza OC1A/OC1B u nisko stanje prilikom brojanja unaprijed, a postavljanje u visoko stanje prilikom brojanja unazad
1	1	Postavljanje izlaza OC1A/OC1B u visoko stanje prilikom brojanja unaprijed, a postavljanje u nisko stanje prilikom brojanja unazad

Ovaj način rada preferira se kod upravljanja elektromotorima. PWM rezolucija definira se ili pomoću ICR1 ili OCR1A registra. Minimalna dozvoljena rezolucija je dva bita tj. ICR1 ili OCR1A registar postavljen u vrijednost $0x0003$, a maksimalna rezolucija je 16-bitova. Za korištenje statične maksimalne vrijednost preporučuje se korištenje ICR1 registra. Frekvencija PWM signala može se izračunati iz slijedeće jednadžbe:

$$f_{OCnxPFCPWM} = \frac{f_{clk-I/O}}{2 \cdot N \cdot TOP} \tag{1}$$

gdje je:

- $f_{OCnxPFCPWM}$ željena frekvencija PWM signala
- $f_{clk-I/O}$ frekvencija takta mikrokontrolera, obično 16 MHz

- N varijabla sa kojom se dijeli frekvencija, a može poprimiti neku od slijedećih vrijednosti: 1, 8, 64, 256 i 1024
- TOP vrijednost do koje će tajmer brojati

5.3 Program za normalni način rada

Funkcija za inicijalizaciju tajmera može se vidjeti u algoritmu 2. Funkcija je definirana kao void jer funkcija ne vraća izlaznu vrijednost. Kao ulazni argument funkcija prima vrijednost do koje želimo da tajmer broji. Tu vrijednost zatim preračunavamo u 16-bitni broj na koji trebamo postaviti tajmer. Nakon toga moramo zabraniti sve prekidne rutine kako bi mogli konfigurirati kontrolne registre TCCR1A, TCCR1B i TCNT1 u koji se sprema vrijednost za unaprijedno postavljanje tajmera te se nakon toga ponovno dozvoljavaju prekidne rutine. Ujedno, postavljanjem vrijednosti 0 u registra TCCR1A odabrali smo normalan način rada pinova OC1A i OC1B tj. odspojeni su od tajmera, a bitove WGM10 i WGM11 postavili smo u nisko stanje dok smo u registar TCCR1B bitove WGM12 i WGM13 postavili u nisko stanje isto kao i bitove za pokretanje tajmera.

```
1  void T1_INIT(uint16_t milisekunde)
2  {
3    T_preset = 65535-(milisekunde*F_CPU/1000/256-1);
4    cli();
5    TCCR1A = 0;
6    TCCR1B = 0;
7
8    TCNT1 = T_preset;
9    sei();
10 }
```

Algoritam 2: Funkcija za inicijalizaciju tajmera

Funkcija za unaprijedno postavljanje tajmera može se vidjeti u algoritmu 3. Ona koristi izračunatu vrijednost za unaprijedno postavljanje tajmera iz algoritma 2 te

tu vrijednost jednostavno upisuje u registra TCNT1.

```
1 void T1_preset()
2 {
3   TCNT1 = T_preset;
4 }
```

Algoritam 3: Funkcija za unaprijedno postavljanje tajmera

Zaustavljanje tajmera vrši se pomoću funkcije T1_stop koja se može vidjeti u algoritmu 4. U funkciji se jednostavno u registar TCCR1B upiše vrijednost 0 što predstavlja prekid brojanja, a u ovom slučaju ne mijenja ni način rada tajmera.

```
1 void T1_stop()
2 {
3   cli();
4   TCCR1B = 0x00;
5   sei();
6 }
```

Algoritam 4: Funkcija za zaustavljanje tajmera

Funkcija T1_start služi za pokretanje tajmera 1, a može se vidjeti u algoritmu 5. Ona u registar TCCR1B upisuje heksadecimalnu vrijednost $0x04$ što znači da smo odabrali dijeljenje unutarnje frekvencije sa 256 i pokrenuli tajmer dok smo u registar TIMSK1 upisali heksadecimalnu vrijednost $0x01$ pomoću koje omogućujemo ulaz u prekidnu rutinu nakon što dođe do preljeva TCNT1 registra.

```
1 void T1_start()
2 {
3   cli();
4   TCCR1B = 0x04;
5   TIMSK1 = 0x01;
6   sei();
```

```
7 }
```

Algoritam 5: Funkcija za pokretanje tajmera

T1_restart funkcija (algoritam 6) služi za ponovno pokretanje tajmera. U funkciji se pozivaju redom funkcije iz algoritama 4, 3 i 5 koje zajedno u kombinaciji čine funkciju za resetiranje tajmera.

```
1 void T1_restart()
2 {
3   T1_stop();
4   T1_preset();
5   T1_start();
6 }
```

Algoritam 6: Funkcija za resetiranje tajmera

Prekidne rutine imaju naziv ISR. Kao ulazni argument primaju vektor prekida koji se može detaljnije vidjeti u ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH [3], a u ovom radu koristi se TIMER1_OVF_vect vektor prekida. U prekidnoj rutini se ponovo unaprijedno postavlja vrijednost TCNT1 registra pomoću funkcije T1_preset (algoritam 3) te se omogućuju prekidi više razine da bi mogli komunicirati sa senzorom i pročitati vrijednost kutne brzine senzora u zadanom trenutku.

```
1 ISR(TIMER1_OVF_vect)
2 {
3   T1_preset();
4   sei();
5
6   MPU_read(MPU_GYRO_XOUT_H, 6, (uint8_t *) &brzina_reg);
7   zamjeni( (int16_t *) &brzina, (uint8_t *) &brzina_reg);
8
9   Integrator();
```

```
10 }
```

Algoritam 7: Prekidna rutina korištena za normalni način rada tajmera

5.4 Program za PWM način rada s korekcijom faze i frekvencije

Funkcija za postavljanje tajmera u PWM način rada može se vidjeti u algoritmu 8. U funkciji se prvo zabranjuju svi prekidi te se zatim TCCR1A registar podešava na način da zapišemo jedinicu na mjesta COM1A1 i COM1B1 bita dok u TCCR1B registar upisujemo jedinice na mjesta WGM13 i CS11 bitova. Nakon toga postavljamo gornju vrijednost do koje tajmer broji i ona iznosi 40000, a upisuje se u ICR1 registar. U registre OCR1A i OCR1B upisujemo vrijednost trajanja visokog dijela signala što nam ujedno predstavlja kut zakreta servo motora.

```
1  void T1_INIT()
2  {
3    cli();
4
5    TCCR1A = 0;
6    TCCR1B = 0;
7
8    TCCR1A |= (1 << COM1A1)|(1 << COM1B1); // ne invertirajuci
         nacin rada
9
10   TCCR1B |= (1 << WGM13) | (1<<CS11);
11   // PWM nacin rada koji koristi ICR1 kao maksimalnanu
         vrijednost brojanja
12
13   ICR1 = 40000;
14
15   OCR1A = 1480;
16   OCR1B = 1400;
```

```
17
18      sei();
19  }
```

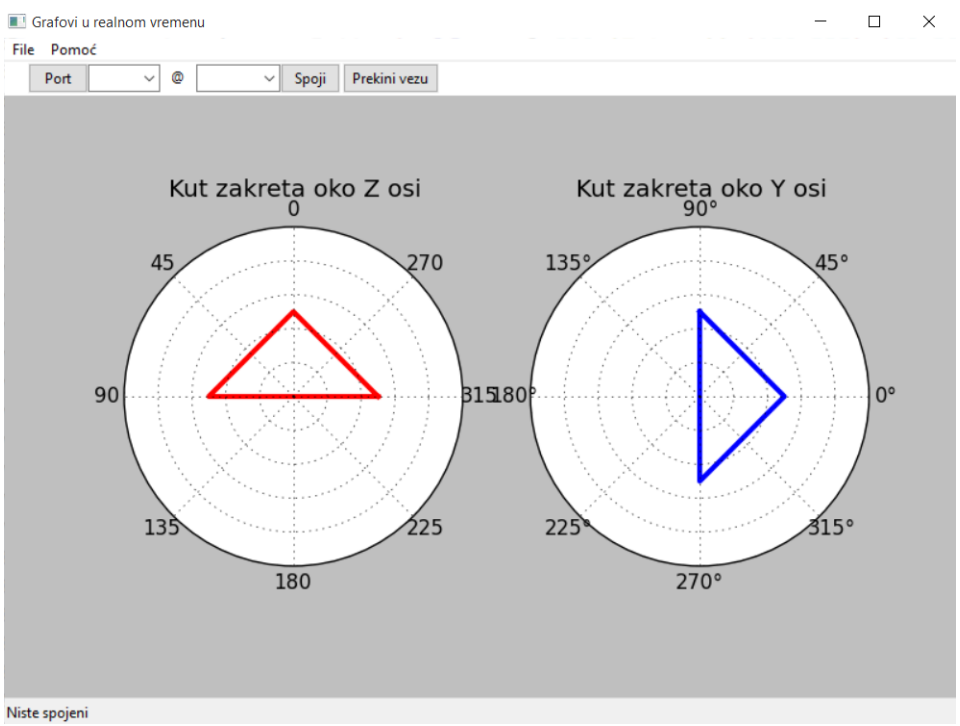
Algoritam 8: Funkcija za postavljanje tajmera u PWM način rada

Registri OCR1A i OCR1B se u daljnjem dijelu programa ažuriraju bez potrebe za zabranom prekida.

6 PROGRAM ZA PRIKAZ PODATAKA NA RAČUNALU

Prva verzija programa za prikazivanje podataka može se vidjeti na slici 31. Program je izrađen u *Python-u* te se sastoji od glavnog izbornika, alatne trake, dva grafa koji se iscrtavaju u realnom vremenu te statusne trake na kojoj se ispisuju obavijesti. Na alatnoj traci nalaze se tipka *Port* pomoću koje se osvježavaju portovi u padajućem izborniku koji je smješten odmah nakon tipke. Nakon portova dolazi padajući izbornik sa kojeg se izabire brzina komunikacije. Nakon što se odabere port i brzina komunikacije klikne se na tipku *Spoji* te se spajamo na port i komuniciramo s uređajem. Na kraju se nalazi tipka za prekid komunikacije.

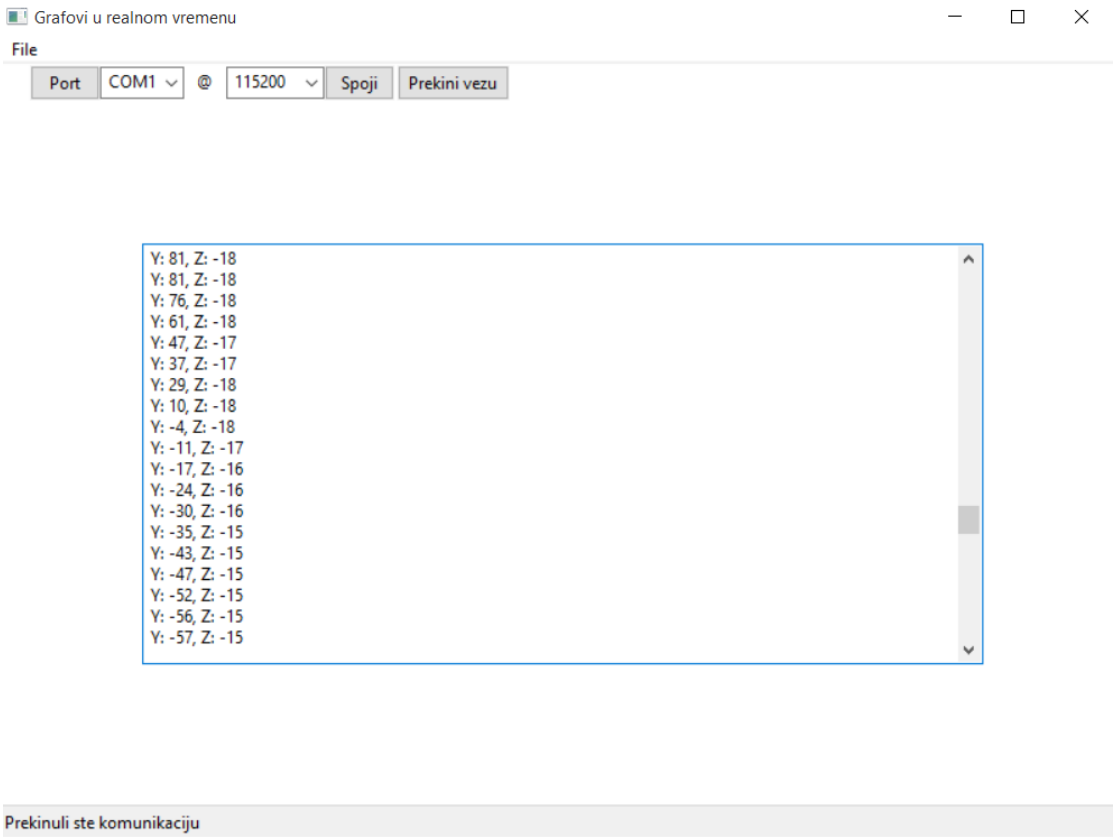
Ova verzija programa nije bila dobra jer je računalo bilo presporo za iscrtavanje grafova u realnom vremenu. Stvaralo se je vrlo veliko kašnjenje te su se grafovi iscrtavali još neko vrijeme nakon prestanka mjerenja kuta zakreta glave.



Slika 31: Prva verzija programa za prikaz podataka o kutu zakreta glave

Konačna verzija programa može se vidjeti na slici 32. Ova verzija programa

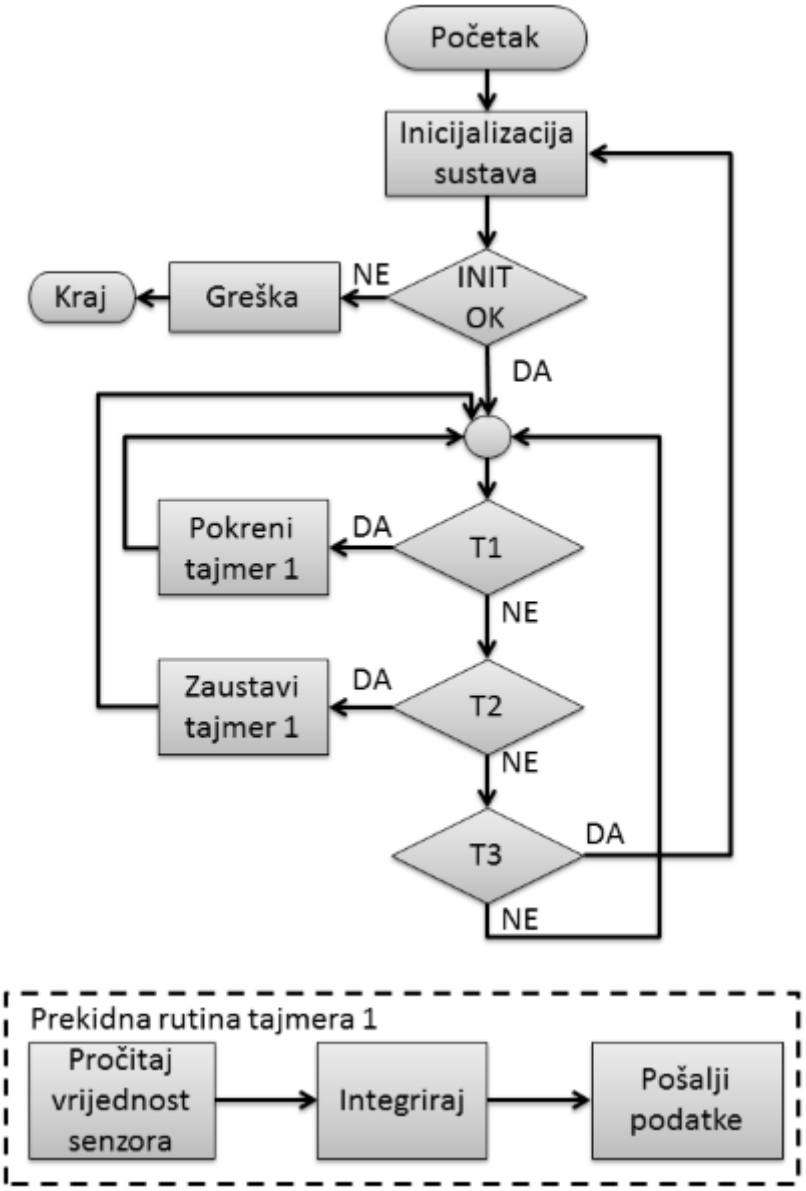
zadržala je istu alatnu i statusnu traku od prijašnje verzije te je umjesto grafova korišteno tekstualno polje za prikaz podataka. Zamjena grafova sa tekstualnim poljem omogućilo je da se podaci prikazuju u realnom vremenu. Podaci se ispisuju na način da se prvo postavi oznaka osi oko koje se mjeri kut te zatim vrijednost izmjerenoga kuta. Orijentacija koordinatnog sustava može se vidjeti na slici 4.



Slika 32: Finalna verzija programa za prikaz podataka o kutu zakreta glave

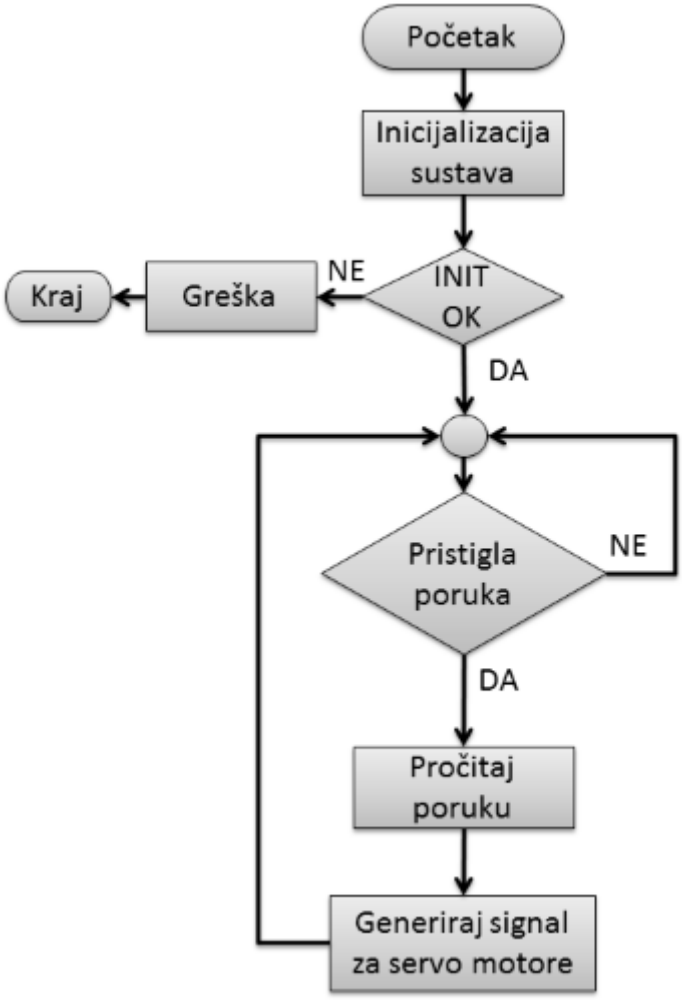
7 PROGRAM ZA UPRAVLJANJE CJELOKUPNIM SUSTAVOM

Na slici 33 možemo vidjeti dijagram toka koda za mjerenje kuta zakreta glave. Inicijalizacija sustava sastoji se od postavlja brzine serijske komunikacije sa bluetoothom, inicijalizacije senzora za mjerenje kuta zakreta glave, postavljanja tajmerskih registara te definiranja pinova za tipke T1, T2 i T3, led diodu i zujalicu.



Slika 33: Dijagram toka koda za mjerenje kuta zakreta glave

Ukoliko je došlo do pogreške prilikom inicijalizacije sustava izvođenje programa se prekida i javlja se signalna zujalica. Nakon inicijalizacije sustava ulazimo u glavnu petlju gdje se stalno provjerava da li je jedna od tri tipke stisnuta. Ako je stisnuta tipka T1 pokreće se tajmer T1 dok tipka T2 zaustavlja tajmer, a tipka T3 ponovo inicijalizira cijeli sustav.
Isto tako na slici 33 možemo vidjeti prekidnu rutinu tajmera 1. U njoj se čitaju vrijednosti kutne brzine koje nam daje senzor te se one integriraju da bi dobili kut zakreta oko osi te se zatim podaci bluetooth-om šalju na drugi sustav.



Slika 34: Dijagram toka koda za upravljanje servo motorima

Sa slike 34 možemo vidjeti dijagram toka koda za upravljanje servo motorima. Kod započinje inicijalizacijom pinova, tajmera za generiranje signala te bluetooth

komunikacije. U slučaju da je došlo do greške prilikom inicijalizacije signalna lampica počinje blicati, a u suprotnome ulazimo u glavnu petlju. U glavnoj petlji se cijelo vrijeme provjerava da li je pristigla poruka preko bluetooth komunikacije te ako je poruka pristigla ona se pročita. Pomoću pročitanih podataka generiraju dva signala za upravljanje servo motorima.

8 RAZMATRANJE ALTERNATIVNOG RJEŠENJA

Umjesto konstruiranja vlastitog uređaja za detekciju orijentacije glave mogli bismo iskoristiti već postojeće uređaje kao što su naočale za virtualnu stvarnost (slika 35) ili pametne mobilne telefone kao što je Samsung koji zajedno u kombinaciji sa Gear VR-om (slika 36) nude mogućnost virtualne stvarnosti.

Slika 35: Naočale za virtualnu stvarnost [18]

Umjesto traženja senzora prikladnih za detektiranje orijentacije glave čovjeka ovi uređaji već dolaze sa ugrađenim žiroskopom te bi bilo potrebno isprogramirati aplikaciju koja bi koristila podatke već ugrađenog žiroskopa i slala ih na sklop za upravljanje kamerom. Ujedno sami uređaji omogućavaju nam i prikaz slike ispred očiju što bi nam dalo stvarni dojam upravljanja mobilnim transportnim uređajem.

Slika 36: Samsung-ov Gear VR [19]

9 PROCJENA TROŠKOVA INVESTICIJA

U tablici 11 može se vidjeti trošak investicije izrađenog sustava koji približno iznosi $1.120,00\ HRK$ dok u tablici 9 možemo vidjeti približni trošak investicije za Samsungov Gear VR sustav koji iznosi $7.770,00\ HRK$, a u tablici 10 vidimo trošak investicije za sustav sa naočalama za virtualnu stvarnost koji iznosi $9.430,00\ HRK$.

Tablica 9: Procjena troškova sustava sa Samsugovim Gear VR sustavom

Naziv komponente	Naziv trgovine	Cijena [HRK]
Samsung Gear VR	Ebay	340
Samsung S7	Ebay	5.100
Nosač kamere	Ebay	110
GoPro HERO4	Ebay	2.220
	Ukupno:	7.770

Tablica 10: Procjena troškova sustava sa naočalama za virtualnu stvarnost

Naziv komponente	Naziv trgovine	Cijena [HRK]
Oculus Rift	Ebay	7.100
Nosač kamere	Ebay	110
GoPro HERO4	Ebay	2.220
	Ukupno:	9.430

Usporedbom podataka iz tablica 9, 10 i 11 možemo vidjeti da je izrađeni sustav puno jeftiniji od sustava koji omogućavaju proširenu stvarnost te je stoga prihvatljiv za skromni studentski budžet.

Tablica 11: Procjena troškova izrađenog sustava

Naziv komponente	Naziv trgovine	Cijena pojedinačno $[HRK]$	Broj komada	Cijena ukupno $[HRK]$
MPU6050	Ebay	19,715	2	39,43
HM10 Bluetooth	Ebay	70,75	2	141,5
ATmega328P-AU	Ebay	10,67	4	42,68
Set smd kondenzatora	Ebay	123,65	1	123,65
FT232RL	Ebay	7,663	4	30,65
Pakovanje oscilatora	Ebay	7,24	1	7,24
Fotopozitivna pločica jednostrana	Elmatis	30	1	30
Fotopozitivna pločica dvostrana	Elmatis	36	2	72
Razvijač	Elmatis	20	1	20
SMD diode	Elmatis	1	10	10
SMD otpornici	Elmatis	0,3	30	9
Stabilizator 5V	Elmatis	5	2	10
Stabilizator 3,3V	Elmatis	12	2	24
Konektori	Elmatis			30
9V baterija	Elmatis	9	6	54
Tipke	Chipoteka	2,65	6	15,9
Nit za 3D printer	Chipoteka	169	2	338
Servo motori	Ebay	29,56	2	59,12
Kamera	Ebay	31,18	2	62,36
			Ukupno:	1.119,53

10 ZAKLJUČAK

U ovom radu izrađena su dva sustava koja zajedno služe za upravljanje kamerom na mobilnom transportnom uređaju. Pomoću prvog sustava možemo vrlo lako odrediti poziciju glave čovjeka, tj. možemo očitati kut zakreta glave oko dvije osi. Sustav se sastoji od kacige, mikrokontrolera, MPU-6050 senzora, bluetooth modula i izvora napajanja. Drugi sustav služi za rotiranje kamere oko dvije osi, a sastoji se od konstrukcije za pomicanje kamere, dva MG996R servo motora, bluetooth modula i izvora napajanja.

Kaciga i konstrukcija za pomicanje kamere izrađene su pomoću 3D printera koji je uvelike ubrzao izradu prototipne kacige i konstrukcije za pomicanje kamere jer su sami 3D printeri napravljeni u svrhu izrade brzih prototipova. Tiskane pločice izrađene su foto postupkom koji je isto razvijem u prototipne svrhe. Te dvije tehnologije omogućile su vrlo jeftinu i brzu izradu cijelog sustava.

Prvi problem javio se je pri izradi sustava kod FT232RL čipa koji je stigao neispravan tj. sa praznom flesh memorijom te ga računalo nije moglo prepoznati i ostvariti komunikaciju i kod servo motora koji su stigli neispravni tj. cijelo su vrijeme vibrirali na mjestu. S bluetooth modulom i MPU-6050 senzorom nije bilo problema isto kao i s Atmega328P mikrokontrolerom.

Drugi problem javio se je kod izrade programa za prikaz podataka. U tom se je slučaju program napravljen u Python-u nedovoljno brzo izvršavao i nije mogao u realnom vremenu iscrtavati grafove koji su trebali prikazivati orijentaciju glave u vertikalnoj i horizontalnoj ravnini, stoga su grafovi zamijenjeni jednostavnim tekstualnim prikazom podataka koji se odvija u realnom vremenu.

Za daljnje unaprjeđenje sustava trebalo bi realizirati jednu od alternativnih ideja koje koriste naočale za virtualnu stvarnost umjesto konstruirane kacige.

LITERATURA

[1] InvenSense Inc.: *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.0*

[2] *HM Bluetooth module datasheet*

[3] Atmel: *ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH*

[4] Future Technology Devices International Ltd.: *FT232R USB UART IC Datasheet Version 2.13*

[5] S. Coros: *Additive Manufacturing*, MIT, predavanja, `http://www.cs.cmu.edu/~scoros/cs15869-s15/lectures/02-3dPrinting.pdf`, 7.11.2016.

[6] `https://en.wikipedia.org/wiki/Fused_deposition_modeling`, 5.11.2016.

[7] `https://ae01.alicdn.com/kf/HTB10aWgKFXXXXb1XFXXq6xXFXXXZ/RC-font-b-Quadcopter-b-font-font-b-GPS-b-font-5-8G-FPV-RTF-4.jpg`, 7.11.2016.

[8] `http://www.mvteamcctv.com/upfile/20141219105125983.jpg`, 1.10.2016.

[9] `http://media.extron.com/img/mktg/environconhumanfact_fig2_3.jpg`, 1.10.2016.

[10] `http://www.sonsivri.to/forum/index.php?topic=59978.0`, 6.11.2016.

[11] `http://i.ebayimg.com/images/g/49IAAOSwal5YGcOG/s-l1600.jpg`, 29.10.2016.

[12] `http://www.hobbyking.com/hobbyking/store/catalog/65384.jpg`, 20.10.2016.

[13] `https://reprappro.com/documentation/mendel-tricolour/`, 7.11.2016.

[14] `http://www.dx.com/p/keyes-hm-10-6-pin-transparent-ble-bluetooth-v4-0-serial-po .WCDWWPkrJPY`, 7.11.2016.

[15] http://ava.upuaut.net/store/image/cache/data/ATmega328TQFP_big-500x500.jpg, 7.11.2016.

[16] https://en.wikipedia.org/wiki/In-system_programming, 7.11.2016.

[17] https://grabcad.com/library/ww-ii-german-paratrooper-fallschirmjager-helmet-1, 7.11.2016.

[18] http://hitdeals.pk/wp-content/uploads/2016/05/VR-BOX-Oculus-Rift-Google-VR-Glass-Cardboard-3D-Virtual-Reality-Glasses-Oculus-jpg, 7.11.2016.

[19] https://cdn.homeshopping.pk/product_images/i/070/samsung_gear_vr_white_press_image__80605_zoom.jpg, 7.11.2016.

PRILOZI

I. Programski kod

II. Tehnička dokumentacija

III. CD-R disc

```
/*    upute za spajanje kontrolera i njegove periferije
 *
 *        tipka 1 -> 2              START
 *        tipka 2 -> 3              STOP
 *        tipka 3 -> 4              RESET
 *
 *        err -> javlja se beeper ->5  BEEPER
 *        ok -> zelena 6            NO_ERR
 *
 *        Softver serial TX -> 9
 *        Softver serial RX -> 8
 *        Bluetooth status  -> 10  STATUS
 */

#include <Wire.h>
#include <SoftwareSerial.h>

// dodavanje vlastitih funkcija
#include "varijable.h"
#include "MPU_funkcije.h"
#include "timer.h"
#include "init_funkcije.h"
#include "bluetooth.h"


void setup()
{

  INIT_SERIAL();       // inicijalizacija serijeske i I2C komuunikacije
  INIT_pins();         // inicijalizacija pinova
  T1_INIT(T);          // inicijalizacija tajmera
  BLUETOOTH_INIT();    // inicijalizacija bluetooth-a
  MPU_INIT();          // inicijalizacija ziroskopa

  if (T1_err() == 1) BEEP();  //Provjera inicijalizacije tajmera
  else ERR = 1;

  if ( MPU_PROVJERA() == 1) BEEP(); //Provjera inicijalizacije ziroskopa
  else ERR = 1;

  s = digitalRead(STATUS); //Provjera inicijalizacije bluetootha
  if (s == 1) BEEP();

  if (ERR == 0) digitalWrite(NO_ERR, HIGH); // sve ok -> nema greske svijetli zeleno
}

```

```
47 ////////////////////////////////////////////////////////////
48
49 //                funkcija za integriranje
50
51 ////////////////////////////////////////////////////////////
52 void Integrator()
53 {
54   Y += T_I*(brzina.y/G_OSJ-G_Y_OFFSET);
55   Z += T_I*(brzina.z/G_OSJ-G_Z_OFFSET);
56
57   if (Y>=90) Y=90;  // ogranicenje kuta oko Y osi
58   else if (Y<-90) Y=-90;
59
60   if (Z>=90) Z=90;  // ogranicenje kuta oko Z osi
61   else if (Z<-90) Z=-90;
62
63   B_serial.print((int) Y);
64   B_serial.print(";");
65   B_serial.print((int) Z);
66   B_serial.println();
67   Serial.print((int) Y);
68   Serial.print(";");
69   Serial.print((int) Z);
70   Serial.println();
71
72 }
73
74 ////////////////////////////////////////////////////////////
75
76 //  interupt
77
78 ////////////////////////////////////////////////////////////
79 ISR(TIMER1_OVF_vect)
80 {
81
82   T1_preset();
83   sei();
84
85   MPU_read(MPU_GYRO_XOUT_H, 6, (uint8_t *) &brzina_reg);
86   zamjeni( (int16_t *) &brzina, (uint8_t *) &brzina_reg);
87   Integrator();
88 }
89
90 ////////////////////////////////////////////////////////////
91
92 //                glavan petlja
93
94 ////////////////////////////////////////////////////////////
95 void loop() {
```

```
96
97    /* ako je doslo do greske prilikom inicijalizacije:
98     *   -> dozvoljava se samo programski reset kontrolera
99     */
100   if (ERR == 1)
101   {
102     digitalWrite(NO_ERR, LOW);
103     tone(BEEPER, 5000);
104
105     if (digitalRead(RESET))
106     {
107       setup();
108       while (digitalRead(RESET)) {}
109     }
110
111   }
112
113   // sav ostali kod ide ovdje
114   else
115   {
116     // start tajmera i odasiljanja podataka
117     if (digitalRead(START))
118     {
119       T1_start();
120       Serial.print("Start");
121       Serial.println();
122       while (digitalRead(START)) {}
123     }
124
125     // stop tajmera i odasiljanja podataka
126     if (digitalRead(STOP))
127     {
128       T1_stop();
129       Y=0;
130       Z=0;
131       B_serial.println("0;0");
132       Serial.print("Stop");
133       Serial.println();
134       while (digitalRead(STOP)) {}
135     }
136
137     // reset kontrolera
138     if (digitalRead(RESET))
139     {
140       digitalWrite(NO_ERR, LOW);
141       setup();
142       while (digitalRead(RESET)) {}
143     }
144
```

```
145       // ako smo ostali bez bluetooth veze
146       if (!digitalRead(STATUS))
147       {
148         tone(BEEPER, 500, 500);
149         BLUETOOTH_CONN();
150       }
151
152     }
153
154 }
```

Algoritam 9: Program za mjerenje kuta zakreta glave

```
1  // ovdje se nalaze sve globalne varijable za program
2
3  ////////////////////////////////////////////////////////////
4  #include <Arduino.h>
5
6  ////////////////////////////////////////////////////////////
7
8  //  definiranje adrese registra koji ce mi trebati za rad
9
10 ////////////////////////////////////////////////////////////
11 // adresa uredjaja za I2C komunikaciju
12 #define MPU 0x68
13
14 // registri za konfiguraciju
15 #define MPU_SMPLRT_DIV 0x19
16 #define MPU_CONFIG 0x1A
17 #define MPU_GYRO_CONFIG 0x1B
18 #define MPU_ACEL_CONFIG 0x1C
19
20 #define MPU_FIFO_EN 0x23
21
22 // registri za citanje vrijednosti ziroskopa
23 #define MPU_ACCEL_XOUT_H 0x3B
24 #define MPU_GYRO_XOUT_H 0x43
25 #define MPU_GYRO_XOUT_L 0x44
26 #define MPU_GYRO_YOUT_H 0x45
27 #define MPU_GYRO_YOUT_L 0x46
28 #define MPU_GYRO_ZOUT_H 0x47
29 #define MPU_GYRO_ZOUT_L 0x48
30
31 // registri za budenje cipa
32 #define MPU_PWR_MGMT_1 0x6B
33 #define MPU_PWR_MGMT_2 0x6C
34
35 #define MPU_FIFO_R_W    0x74
36 #define MPU_WHO_AM_I    0x75
37
38 ////////////////////////////////////////////////////////////
39
40 //  definiranje vlastite strukture varijabli i konstanti
41
42 ////////////////////////////////////////////////////////////
43 #define G_OSJ 131
44 #define A_OSJ 16384
45 #define G_X_OFFSET 3
46 #define G_Y_OFFSET -2
47 #define G_Z_OFFSET 0
48
49
```

```
50 struct BRZINA_REG
51 {
52   uint8_t kut_x_h;
53   uint8_t kut_x_l;
54   uint8_t kut_y_h;
55   uint8_t kut_y_l;
56   uint8_t kut_z_h;
57   uint8_t kut_z_l;
58 }brzina_reg;
59
60
61 struct BRZINA
62 {
63   int16_t x;
64   int16_t y;
65   int16_t z;
66 }brzina;
67
68 struct KUT
69 {
70   float x;
71   float y;
72   float z;
73 }kut;
74
75 static float Y, Z = 0.0;
76
77 ////////////////////////////////////////////////////////////
78
79 //   definiranje vremenske konstante i varijable tajmera
80
81 ////////////////////////////////////////////////////////////
82 #define T 100        // u milisekundama
83 #define T_I 0.1      // u sekundama
84 uint16_t T_preset;
85
86
87 ////////////////////////////////////////////////////////////
88
89 //                  definiranje pinova
90
91 ////////////////////////////////////////////////////////////
92
93 #define START 15  // analogni pinovi 14+ pin na koji je spojeno
94 #define STOP 16
95 #define RESET 14
96
97 #define BEEPER 7
98 #define NO_ERR 6
```

```
99
100 #define TX 9
101 #define RX 8
102 #define STATUS 10
103
104 int s = 0;
105 int ERR=0;
```

Algoritam 10: Varijable

```
1  // ovdje se nalaze sve funkcije vezane za rad s MPU6050 cipom
2
3  ////////////////////////////////////////////////////////////
4  #include <Arduino.h>
5
6
7  ////////////////////////////////////////////////////////////
8
9  //            FUNKCIJA ZA CITANJE IZ REGISTRA
10
11 ////////////////////////////////////////////////////////////
12 void MPU_read(int registar, int broj_registara, uint8_t *var)
13 {
14   /*
15    * Funkcija cita zadan broj registar, a kao ulazne argumet
         prima
16    * pocetni registar, broj koji definira koliko registara
         zelimo
17    * procitati i pojnter varijablu u koju vraca podatke
18    */
19   int br_reg_s;
20   Wire.beginTransmission(MPU);
21   Wire.write(registar);       // registar za citanje
22   Wire.endTransmission(false);
23
24   Wire.requestFrom(MPU, broj_registara, true);
25
26   br_reg_s = 0;
27   while (Wire.available() && br_reg_s < broj_registara)
28   {
29     var[br_reg_s] = Wire.read();
30     br_reg_s++;
31   }
32 }
33 ////////////////////////////////////////////////////////////
34
35 //            FUNKCIJA ZA PISANJE U REGISTAR
36
37 ////////////////////////////////////////////////////////////
38 void MPU_write(int adresa_reg, uint8_t podatci)
39 {
40   /*
41    * Funkcija kao ulazni argument prima adresu registra
42    * u koji se zeli upisati podata te 8 bitni podatak
43    * koji se upisuje u registar
44    */
45   Wire.beginTransmission(MPU);
46   Wire.write(adresa_reg);              // adresa registra
47   Wire.write(podatci);            // podatci za upis u registar
```

```
48    Wire.endTransmission(true);
49  }
50  ////////////////////////////////////////////////////////////
51
52  //            FUNKCIJA ZA INICIJALIZACIJU SENZORA
53
54  ////////////////////////////////////////////////////////////
55  void MPU_INIT()
56  {
57    MPU_write(MPU_PWR_MGMT_1, 0x08); // bilo je 9 umjesto 8
58
59    // onemogucuje mjerenja akceleracije
60    MPU_write(MPU_PWR_MGMT_2, 0x38);
61
62    // konfiguracija low pass filtera (DLPF)
63    // 0x03 -> max delay 4.8ms, Fs = 1kHz, bandwidth = 42Hz
64    MPU_write(MPU_CONFIG, 0x03);
65    MPU_write(MPU_SMPLRT_DIV, bit(6));
66
67    //konfiguracija ziroskopa
68    MPU_write(MPU_GYRO_CONFIG, 0x00);
69  //  MPU_write(MPU_ACEL_CONFIG, 0x00);
70
71  }
72
73  int MPU_PROVJERA()
74  {
75    uint8_t c;
76    int ok = 1;
77
78    MPU_read (MPU_WHO_AM_I, 1, &c);
79    if ( c != 0x68) ok = 0;
80    //  else  return 0;
81
82    //  MPU_read (MPU_PWR_MGMT_2, 1, &c);
83    //  if ( c != 0x38) ok = 0;
84    //  else  return 0;
85
86    MPU_read (MPU_GYRO_CONFIG, 1, &c);
87    if ( c != 0x0) ok = 0;
88    //  else  return 0;
89
90    return ok;
91  }
```

Algoritam 11: MPU funkcije

```
1  // funkcija za rad s tajmerima
2
3  ////////////////////////////////////////////////////////////
4  #include <Arduino.h>
5
6  ////////////////////////////////////////////////////////////
7  void T1_INIT(uint16_t milisekunde)
8  {
9    /* fukncija kao ulaznu varijablu prima vijednost
10    * u milisekundama te zatim racuna vrijednost na
11    * koju treba postaviti tajmer da broji tocno
12    * odredeno vrijeme
13    */
14   T_preset = 65535-(milisekunde*F_CPU/1000/256-1);
15   cli();
16   TCCR1A = 0;      // postavljanje kontrolnih registara na nulu
17   TCCR1B = 0;
18
19   TCNT1 = T_preset; // preset vrijednosti tajmera
20   sei();
21   delay(100);
22 }
23
24 void T1_preset()
25 {
26   TCNT1 = T_preset;
27 }
28
29
30 void T1_stop()
31 {
32   cli();
33   TCCR1B = 0x00;  // kad je nula, tajmer ne broji
34   sei();
35 }
36
37
38 void T1_start()
39 {
40   cli();
41   TCCR1B = 0x04; // clock / 256
42   TIMSK1 = 0x01; // dozvoli prekid
43   sei();
44 }
45
46
47 void T1_restart()
48 {
49   T1_stop();
```

```
50    T1_preset();
51    T1_start();
52  }
53
54
55  unsigned long T1_read()
56  {
57    // funkcija cita trenutnu vrijednost tajmera
58    uint16_t tcnt;
59    cli();
60    tcnt = TCNT1;
61    sei();
62    return tcnt;
63  }
64
65
66  int T1_err()
67  {
68    /* funkcija vraca nulu ako je doslo do
69     * greske prilikom unosenja preset vrijednosti
70     */
71    if (T1_read()!=T_preset) return 0;
72    else return 1;
73  }
```

Algoritam 12: Funkcije za tajmer

```
1  // ovdje se nalaze sve za inicijalizaciju kontrolera
2
3  ////////////////////////////////////////////////////////////
4  #include <Arduino.h>
5  ////////////////////////////////////////////////////////////
6  void INIT_SERIAL()
7  {
8    Serial.begin(115200);
9    Wire.begin();
10 }
11
12 void INIT_pins()
13 {
14   pinMode(BEEPER, OUTPUT);
15   pinMode(NO_ERR, OUTPUT);
16   pinMode(START, INPUT);
17   pinMode(STOP, INPUT);
18   pinMode(RESET, INPUT);
19   pinMode(STATUS, INPUT);
20 }
21
22 void BEEP()
23 {
24   digitalWrite(BEEPER, HIGH);
25   delay(100);
26   digitalWrite(BEEPER, LOW);
27   delay(100);
28 }
29
30 void BLINK()
31 {
32   digitalWrite(NO_ERR, HIGH);
33   delay(100);
34   digitalWrite(NO_ERR, LOW);
35   delay(100);
36 }
37 ////////////////////////////////////////////////////////////
38
39 //          FUNKCIJA ZA ZAMJENU MJESTA REGISTARA
40
41 ////////////////////////////////////////////////////////////
42 void zamjeni(int16_t *u_sto, uint8_t *sto)
43 {
44   u_sto[0]=(sto[0]*256+sto[1]);
45   u_sto[1]=(sto[2]*256+sto[3]);
46   u_sto[2]=(sto[4]*256+sto[5]);
47 }
```

Algoritam 13: Inicijalizacijske funkcije

```
1  // ovdje se nalaze funkcije za kontrolu bluetootha
2
3  ////////////////////////////////////////////////////////
4  #include <Arduino.h>
5
6  SoftwareSerial B_serial(RX, TX); // RX 9, TX 8
7
8  ////////////////////////////////////////////////////////
9  void BLUETOOTH_CONN()
10 {
11   B_serial.print("AT+CON74DAEAB31F33");  // spajanje na drugi
       uredaj
12   delay(1500);
13 }
14
15 void BLUETOOTH_INIT()
16 {
17   s = 0;
18   B_serial.begin(9600);
19   BLUETOOTH_CONN();
20 }
```

Algoritam 14: Bluetooth funkcije

```
1  #!/usr/bin/env python
2  # -*- coding: utf_8 -*-
3  import wx
4  from komunikacija import COM
5
6  _sirina = 800
7  _visina = 600
8  _milisekunde = 10
9
10 #-----------------------------------------------
11
12 #               Glavni prozor
13
14 #-----------------------------------------------
15 class Glavni_prozor(wx.Frame):
16
17     def __init__(self):
18         """Inicijalizacija"""
19         wx.Frame.__init__(self,parent=None,title='Grafovi u
               realnom vremenu', size=(_sirina,_visina),
20                           style=wx.DEFAULT_FRAME_STYLE & ~wx.
                                 RESIZE_BORDER)
21         self.Center()
22         self.statusbar = self.CreateStatusBar()
23         self.statusbar.SetStatusText('Niste spojeni')
24
25         #       definiranje glavnog panela
26         self.panel = wx.Panel(self,-1)
27         self.SetBackgroundColour('white')
28
29         #       definiranje panela za grafove
30         self.g_panel = wx.Panel(self.panel,-1, pos=(0,28),size=(
               _sirina,500))
31         self.g_panel.SetBackgroundColour('white')
32         #       definiranje izbornika
33         menu=wx.MenuBar()
34         FILE=wx.Menu()
35         FILE.Append(wx.ID_EXIT, 'Zatvori', 'Zatvori program')
36         menu.Append(FILE,'File')
37         self.SetMenuBar(menu)
38
39         #       povezivanje funkcije s izbornikom
40         self.Bind(wx.EVT_MENU, self.Izadi, id=wx.ID_EXIT)
41         self.Bind(wx.EVT_CLOSE, self.Izadi)
42
43         #       definiranje tajmera
44         self.timer = wx.Timer(self, id=10)
45         self.Bind(wx.EVT_TIMER, self.Citaj, id=10)
46
```

```
47      #-----------------------------------------------
48          # postavljanje alatne trake
49          alatna_traka = Alatna_traka(self.panel, self.statusbar,
                self.timer)
50
51          self.textbox = wx.TextCtrl(self.g_panel, id=150, pos
                =(100,100), size=(_sirina -2*100,300),
52                                     style=wx.TE_READONLY | wx.
                                           TE_MULTILINE)
53
54          h_box = wx.BoxSizer()
55          h_box.Add(alatna_traka, 1, wx.EXPAND | wx.ALL, 5)
56          h_box.Add(self.textbox, 1, wx.EXPAND | wx.ALL, 5)
57          self.panel.SetSizer(h_box)
58
59      def Citaj(self,event):
60
61          procitano = COM.read().split(",")    # djeljenje
                procitanih podatka u listu
62          try:
63              z = int(procitano[1])
64              y = int(procitano[0])
65              self.textbox.AppendText("Y: %i, Z: %i\n"%(y,z))
66          except:
67              pass
68  ##              z=0
69  ##              y=0
70
71  ##          self.textbox.AppendText("Y: %i, Z: %i\n"%(y,z))
72
73      #-----------------------------------------------
74
75      #           funkcija menu bara
76
77      #-----------------------------------------------
78      def Izadi(self,event):
79          """Funkcija za gasenje prozora"""
80          self.timer.Stop()
81          COM.close()
82
83          self.Destroy()
84          event.Skip()
85
86  #-------------------------------------------------
87
88  #               Alatna traka
89
90  #-------------------------------------------------
91  class Alatna_traka(wx.Panel):
```

```
92
93      def __init__(self,parent, statusbar, timer):
94          wx.Panel.__init__(self,parent,-1, size=(_sirina,28), pos
                =(0,0))
95          self.SetBackgroundColour('white')
96          self.statusbar = statusbar
97          self.timer = timer
98
99          #    definiranje gumbiju i padajucih izbornika
100         g_port = wx.Button(self,100,'Port',(20,1),(50,25))
101         self.p_serijska = wx.ComboBox(self,105,'',(70,2),(60,25)
                ,COM.available())
102         tekst = wx.StaticText(self,-1,'@',(135,5),(20,25),wx.
                ALIGN_CENTER)
103         p_bdrate = wx.ComboBox(self, 110, '', (160,2),(70,25),['
                2400', '9600', '19200', '38400', '115200', '250000'])
104         g_connect = wx.Button(self, 115, 'Spoji',(230,1),(50,25)
                )
105         g_disconnect = wx.Button(self, 120, 'Prekini vezu'
                ,(282,1),(80,25))
106
107         # povezivanje padajucih izbornika i tipki
108         self.Bind(wx.EVT_BUTTON, self.Refresh_port, id=100)
109         self.Bind(wx.EVT_BUTTON, self.Spoji, id=115)
110         self.Bind(wx.EVT_BUTTON, self.Odspoji, id=120)
111         self.Bind(wx.EVT_COMBOBOX, self.P_serijska, id=105)
112         self.Bind(wx.EVT_COMBOBOX, self.P_bdrate, id=110)
113
114     #-----------------------------------------------
115
116     #       funkcije alatne trake
117
118     #-----------------------------------------------
119     def Refresh_port(self,event):
120         """Ponovo pretrazuje portove"""
121         self.p_serijska.SetItems(COM.available())
122
123     def Spoji(self,event):
124         """Spajanje serijske komunikacije"""
125         try:
126             if self.port and self.bdrate:
127                 c = COM.connect(self.port,eval(self.bdrate))
128                 if c is True:
129                     self.statusbar.SetStatusText(u'Spojeno na %s
                            @ %s\n'%(self.port,self.bdrate))
130                     self.timer.Start(_milisekunde)
131                 else:
132                     self.statusbar.SetStatusText(u'Niste se
                            uspjeli spojiti na port %s\n'%self.port)
```

```
133             except:
134                 self.statusbar.SetStatusText(u'Niste odabrali port i
                        /ili brzinu serijske veze\n')
135
136         def Odspoji(self,event):
137             COM.close()
138             self.timer.Stop()
139             self.statusbar.SetStatusText(u'Prekinuli ste
                    komunikaciju\n')
140
141         def P_serijska(self,event):
142             self.port=event.GetString()
143         def P_bdrate(self,event):
144             self.bdrate=event.GetString()
145
146 #---------------------------------------------
147
148 #           Pokretanje aplikacije
149
150 #---------------------------------------------
151 if __name__ == '__main__':
152     aplikacija=wx.App(0)
153     grafovi=Glavni_prozor()
154     grafovi.Show()
155     aplikacija.MainLoop()
```

Algoritam 15: Grafičko sučelje

```
#!/usr/bin/env python
# -*- coding: utf_8
"""Klasa za serijsku komunikaciju"""

import serial
from serial.tools import list_ports

#--------------------------------------------

#        Klasa za serijsku komunikaciju

#--------------------------------------------
class COM():
    ard = None

    @classmethod
    def available(clc):
        return [i[0] for i in list_ports.comports()]

    @classmethod
    def connect(clc,port,bdrate):
        try:
            clc.ard=serial.Serial(port,bdrate,timeout=1)
            return True
        except:
            return False

    @classmethod
    def close(clc):
        try:
            clc.ard.close()
        except:
            pass

    @classmethod
    def read(clc):
        return clc.ard.readline(clc.ard.inWaiting())
```

Algoritam 16: Klasa za komunikaciju

```
/*    Upute za spajanje
 *
 *    Servo motor
 *    Smeda - gnd
 *    crvena - 5V
 *    zuta - signal
 *
 *    LED -> pin 3
 *
 *    Softver serial TX -> 5
 *    Softver serial RX -> 6
 *    Bluetooth status  -> 7  STATUS
 */

#include <SoftwareSerial.h>

// dodavanje vlastitih funkcija
#include "varijable.h"
#include "timer.h"


// inicijalizacija bluetooth komunikacije
SoftwareSerial B_serial(RX, TX);

String procitano = "";

int kut_z;
int kut_y;

void setup() {

  Serial.begin(9600);
  B_serial.begin(9600);
  pinMode(P_S1, OUTPUT);
  pinMode(P_S2, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(STATUS, INPUT);

  T1_INIT();
  delay(200);
  S2 = kut_gornji_do(0);
  S1 = kut_donji_do(0);

  digitalWrite(LED, HIGH);

}


void loop() {
```

```
50
51      if (B_serial.available())
52      {
53        digitalWrite(LED, HIGH);
54        int poruka = B_serial.read();
55        if (isDigit(poruka))
56        {
57          procitano += (char)poruka;
58        }
59
60        else if (poruka == '-'){
61          procitano += (char)poruka;
62        }
63
64        else if (poruka == ';'){  // prvi kut
65          kut_z = procitano.toInt();
66          procitano ="";
67        }
68
69        else if (poruka == '\n')
70        {
71          kut_y = procitano.toInt();
72
73          if (kut_y<0)
74          {
75            S1 = kut_donji_do(kut_y);
76          }
77          else
78          {
79            S1 = kut_donji_od(kut_y);
80          }
81
82          if (kut_z<0)
83          {
84            S2 = kut_gornji_do(kut_z);
85          }
86          else
87          {
88            S2 = kut_gornji_od(kut_z);
89          }
90
91          procitano ="";
92        }
93  //      if (digitalRead(LED)){
94  //        digitalWrite(LED, LOW);
95  //      }
96  //      else { digitalWrite(LED, HIGH); }
97  //      digitalWrite(LED, !digitalRead(LED));
98      }
```

```
99
100 }
```

Algoritam 17: Program za upravljanje kamerom

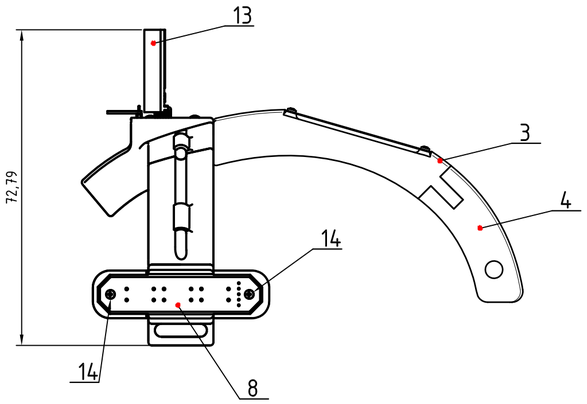
```
#ifndef TIMER_H
#define TIMER_H
// funkcija za rad s tajmerima

//////////////////////////////////////////////////////////////
#include <Arduino.h>

//////////////////////////////////////////////////////////////
void T1_INIT()
{
  cli();

  TCCR1A = 0;
  TCCR1B = 0;

  TCCR1A |= (1 << COM1A1)|(1 << COM1B1); // ne invertirajuci
      nacin rada

  TCCR1B |= (1 << WGM13) | (1<<CS11);
  // PWM nacin rada koji koristi ICR1 kao maksimalnanu
      vrijednost brojanja

  ICR1 = 40000;

  OCR1A = 1480;
  OCR1B = 1400;
  sei();
}


void T1_stop()
{
  TCCR1B = 0x00;  // kad je nula, tajmer ne broji
}


void T1_start()
{
  cli();
  TCCR1A |= (1 << COM1A1)|(1 << COM1B1);
  TCCR1B |= (1 << WGM13) | (1<<CS11);
  sei();
}


void T1_restart()
{
  T1_stop();
  T1_start();
```

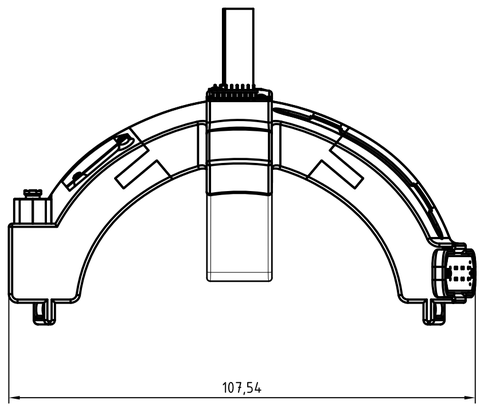
```
48 }
49
50 #endif
```

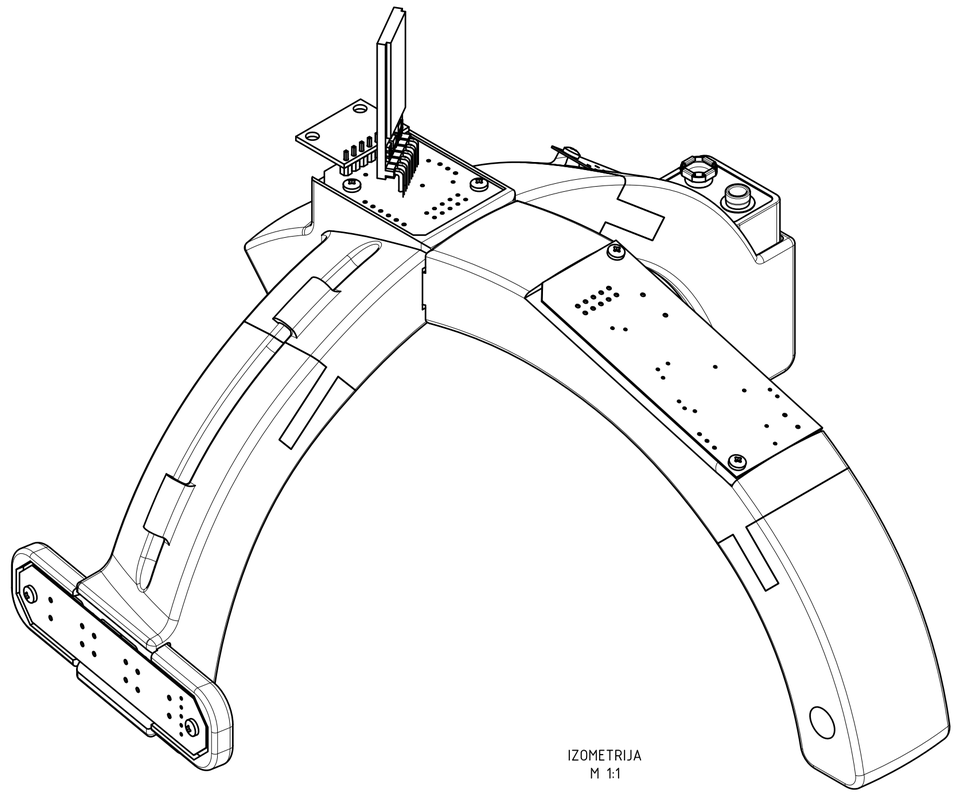
Algoritam 18: Funkcija za podešavanje tajmera

```
#ifndef VARIJABLE_H
#define VARIJABLE_H
////////////////////////////////////////////////

//      Definiranje potrebnih varijabli

////////////////////////////////////////////////
#include <Arduino.h>


// donji motor mapiranje kuta u vrijednost registra
#define kut_donji_do(x) (540+(x+90.)*((1480.-540)/(0+90)))
#define kut_donji_od(x) (1480+(x+0.)*((2500.-1480)/(90-0)))

// gornji motor mapiranje kuta u vrijednost registra
#define kut_gornji_do(x) (2500+(x+90.)*((1490.-2500)/(90-0)))
#define kut_gornji_od(x) (1490+(x+0.)*((565.-1490)/(90-0)))


// definiranje varijabli
#define P_S1 9
#define P_S2 10
#define S1 OCR1A
#define S2 OCR1B


#define TX 5
#define RX 6
#define STATUS 7

#define LED 3


#endif
```

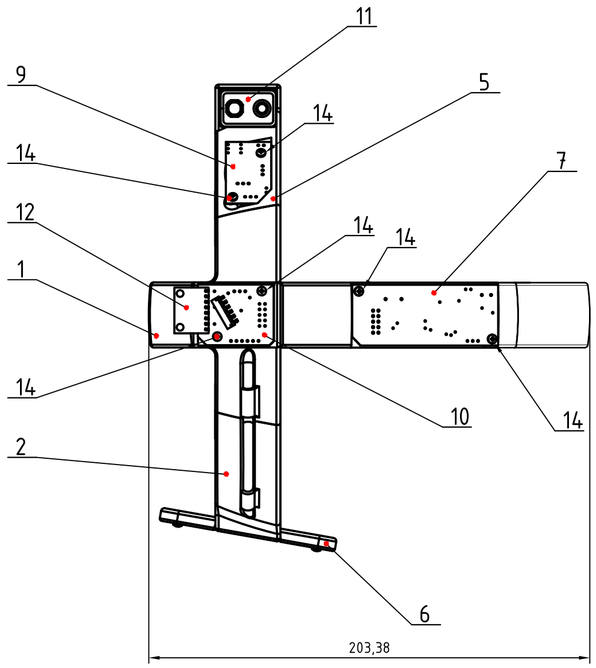
Algoritam 19: Varijable







IZOMETRIJA
M 1:1



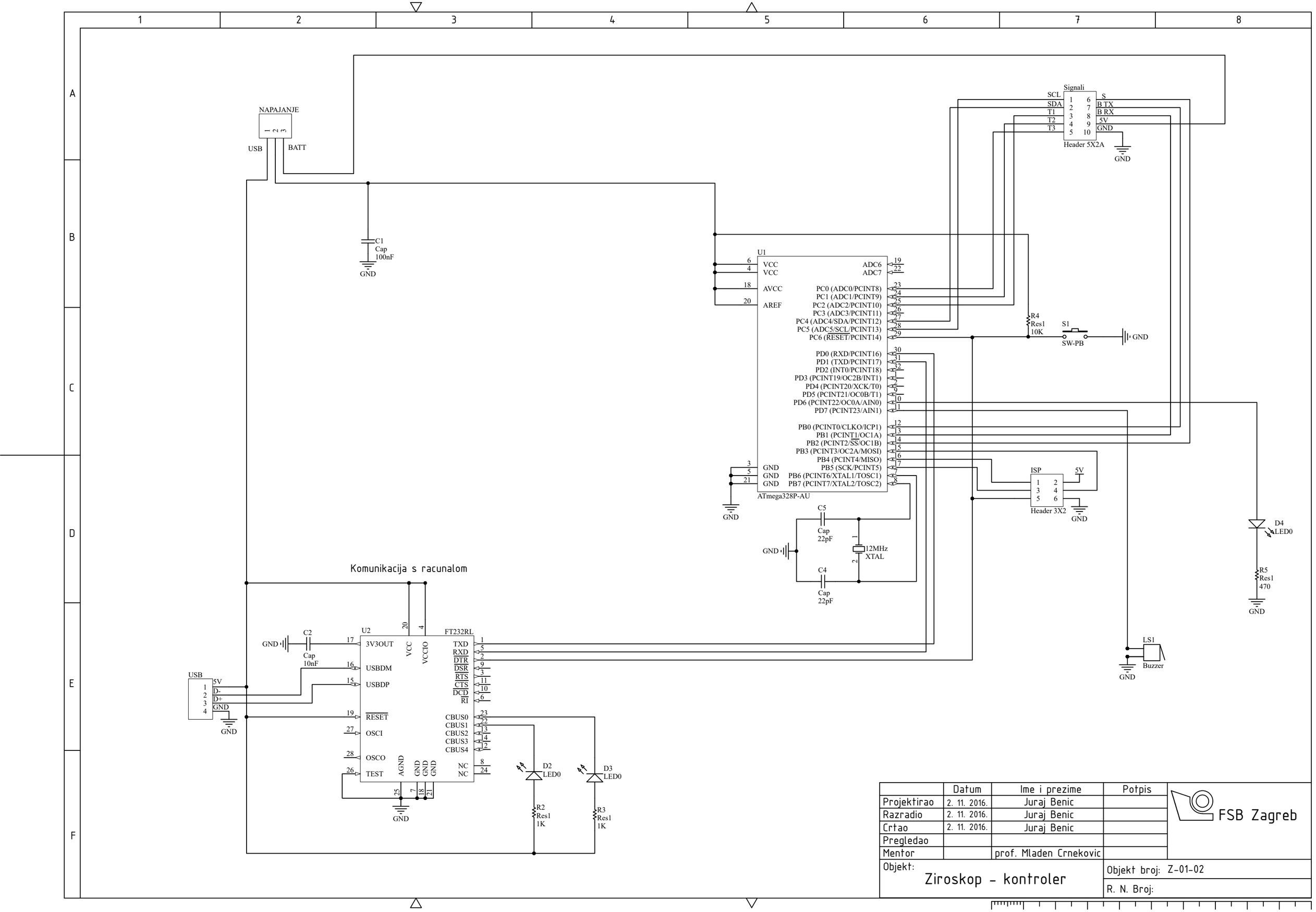
Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
14	Vijak ST2.2x4.5-C-H	8	ISO 7049	Čelik		0.17 g
13	Bluetooth modul	1				
12	MPU 6050	1				
11	9V baterija	1			48x26x17	
10	Ziroskop - periferija	1	Z-04-00	Vitroplast	35x28x1,5	
9	Ziroskop - napajanje	1	Z-03-00	Vitroplast	37x21x1,5	
8	Ziroskop - tipke	1	Z-02-01	Vitroplast	75x16x1,5	
7	Ziroskop - kontroler	1	Z-01-02	Vitroplast	70x28x1,5	
6	Nosač tipki	1		PLA		
5	Zadnji donji dio kacige	1		PLA		
4	Zadnji gornji dio kacige	1		PLA		
3	Prednji lijevi dio kacige	1		PLA		
2	Prednji desni dio kacige	1		PLA		
1	Centralni dio kacige	1		PLA		

Broj naziva - code		Datum	Ime i prezime	Potpis
	Projektirao	8.1.2017.	Juraj Benic	
	Razradio	8.1.2017.	Juraj Benic	
	Crtao	8.1.2017.	Juraj Benic	
	Pregledao			

FSB Zagreb

ISO - tolerancije	Objekt:	Objekt broj:
		R. N. broj:
	Napomena:	Kopija
	Materijal: / Masa:	
Mjerilo originala 1:2	Naziv: SKLOP ZA MJERENJE KUTA ZAKRETA GLAVE	Pozicija:
	Crtež broj: Z-00-01	Format: A2
		Listova:
		List: 1

0 10 20 30 40 50 60 70 80 90 100



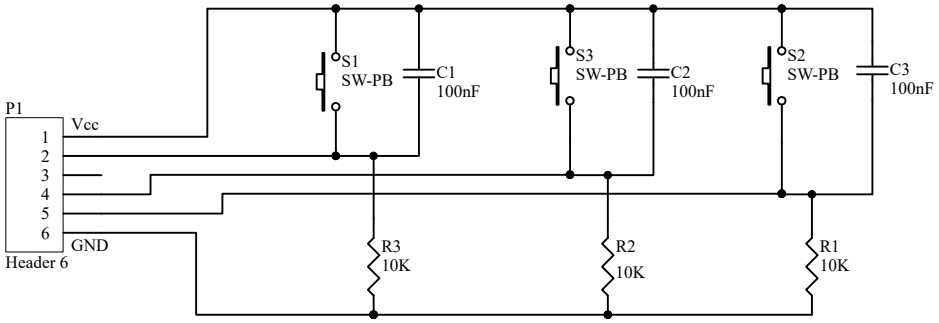
	Datum	Ime i prezime	Potpis
Projektirao	2. 11. 2016.	Juraj Benic	
Razradio	2. 11. 2016.	Juraj Benic	
Crtao	2. 11. 2016.	Juraj Benic	
Pregledao			
Mentor		prof. Mladen Crnekovic	

FSB Zagreb

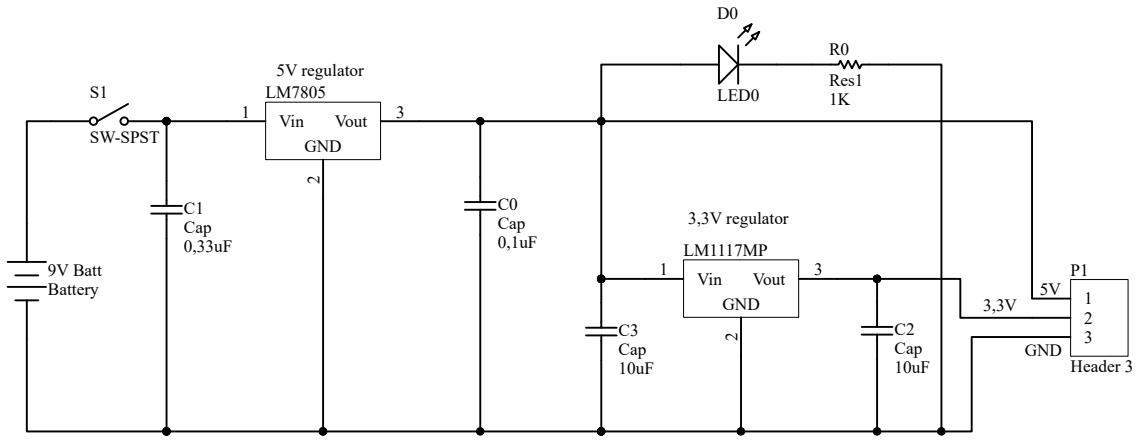
Objekt: Ziroskop - kontroler

Objekt broj: Z-01-02

R. N. Broj:



	Datum	Ime i prezime	Potpis	
Projektirao	2. 11. 2016.	Juraj Benic		FSB Zagreb
Razradio	2. 11. 2016.	Juraj Benic		
Crtao	2. 11. 2016.	Juraj Benic		
Pregledao				
Mentor		prof. Mladen Crnekovic		
Objekt: Ziroskop - tipke			Objekt broj: Z-02-01	
			R. N. Broj:	



	Datum	Ime i prezime	Potpis
Projektirao	2. 11. 2016.	Juraj Benic	
Razradio	2. 11. 2016.	Juraj Benic	
Crtao	2. 11. 2016.	Juraj Benic	
Pregledao			
Mentor		prof. Mladen Crnekovic	

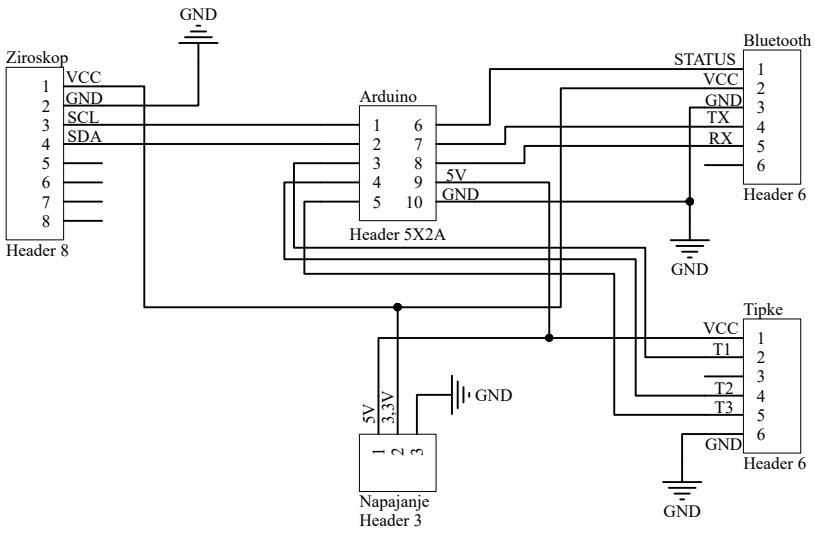
FSB Zagreb

Objekt:

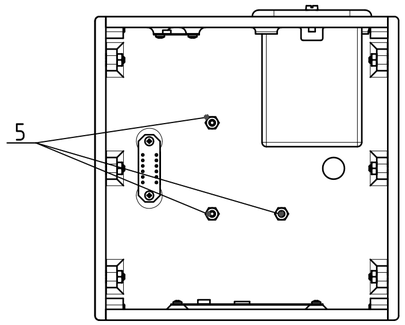
Ziroskop - napajanje

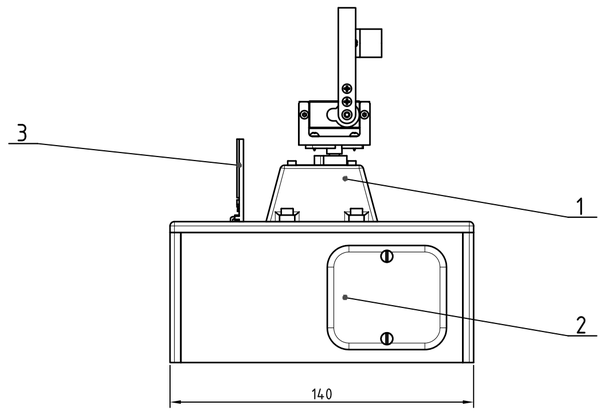
Objekt broj: Z-03-00

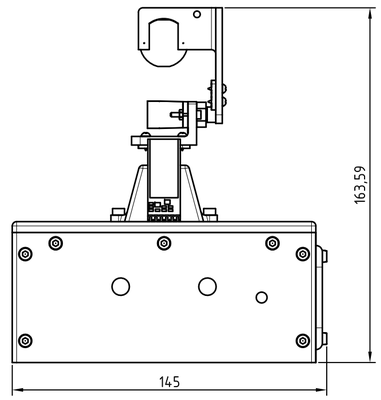
R. N. Broj:

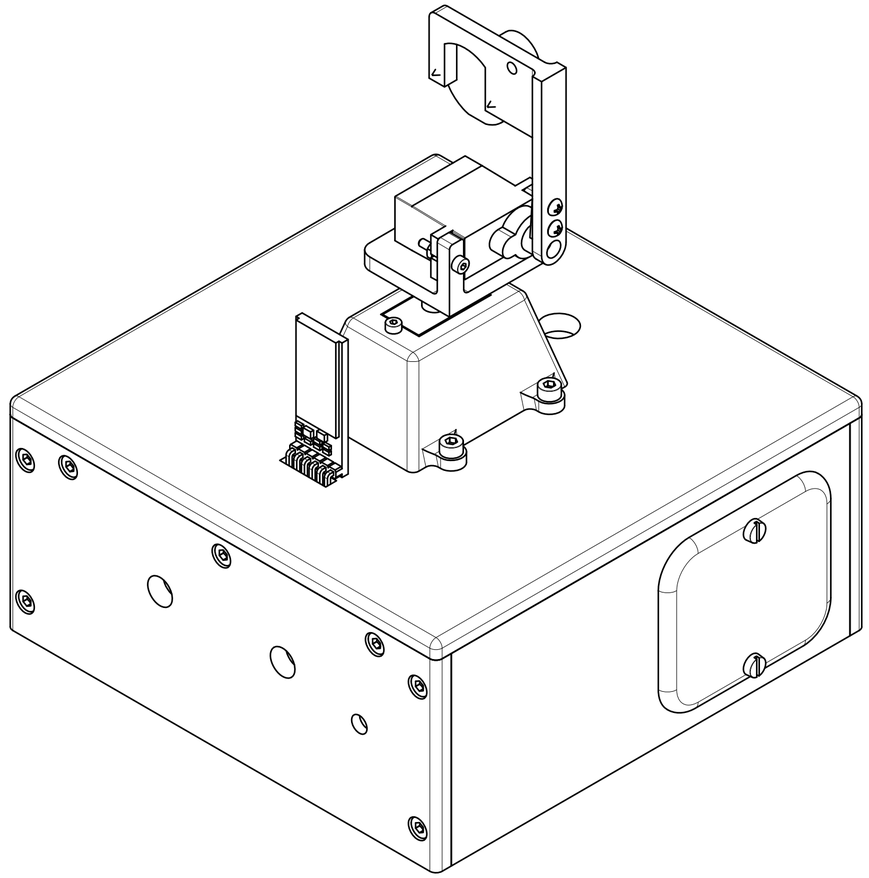


	Datum	Ime i prezime	Potpis	
Projektirao	2. 11. 2016.	Juraj Benic		FSB Zagreb
Razradio	2. 11. 2016.	Juraj Benic		
Crtao	2. 11. 2016.	Juraj Benic		
Pregledao				
Mentor		prof. Mladen Crnekovic		
Objekt: Ziroskop - periferija			Objekt broj: Z-04-00	
			R. N. Broj:	

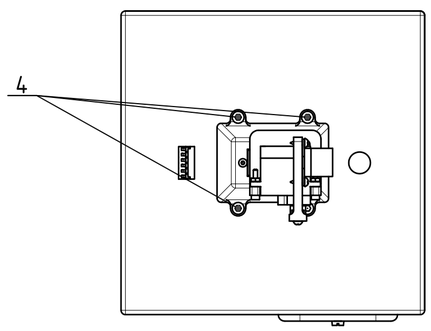








IZOMETRIJA
M1:1



Poz.	Naziv dijela	Kom.	Crtež broj / Norma	Materijal	Sirove dimenzije / Proizvođač	Masa
5	Matica M3	4	ISO 4032			0,051 g
4	Vijak M3x16	4	ISO 4762			0,17 g
3	HM-10 bluetooth	1				
2	Sklop kutije	1	KU-000-001			218 g
1	Sustav za pomicanje kamere	1	SK-000-002			62 g

Broj naziva - code		Datum	Ime i prezime	Potpis
	Projektirao	10.1.2017.	Juraj Benic	
	Razradio	10.1.2017.	Juraj Benic	
	Crtao	10.1.2017.	Juraj Benic	
	Pregledao			

FSB Zagreb

ISO - tolerancije

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Materijal:

Masa: 282 g

Mjerilo originala 1:2

Naziv: SKLOP ZA POMICANJE KAMERE

Pozicija:

Format: A2

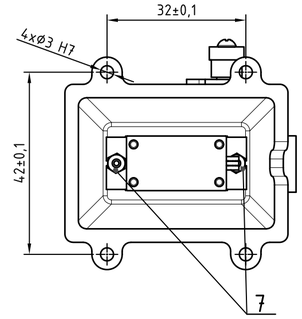
Listova: 13

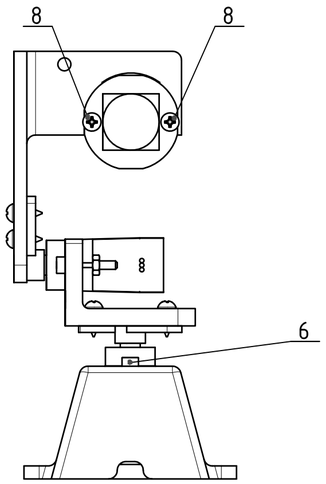
Crtež broj: S-000-001

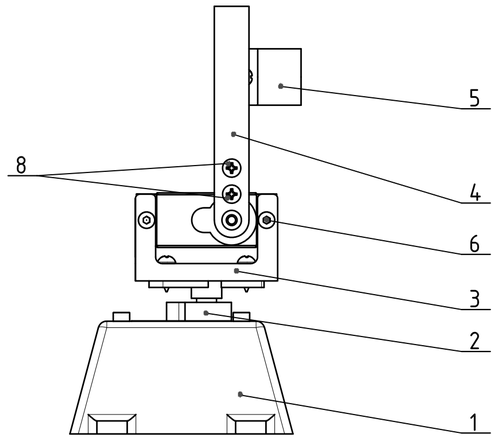
List: 1

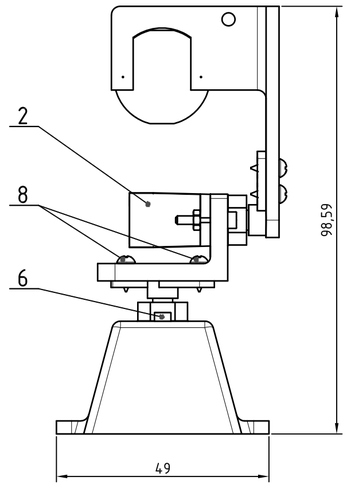
Design by CADLab

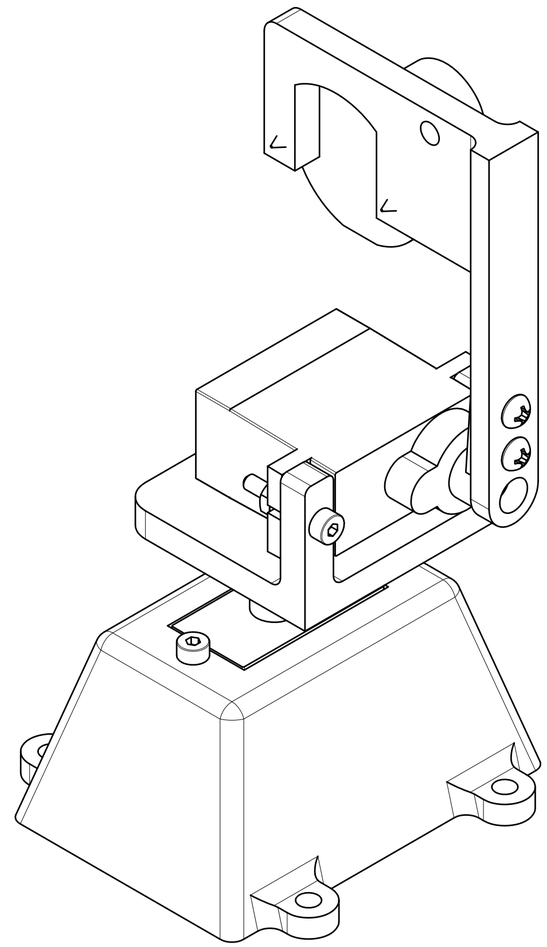
0 10 20 30 40 50 60 70 80 90 100



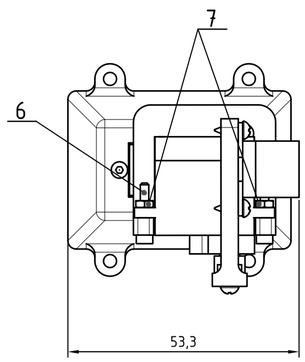








Izometrija
M2:1



Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
8	Vijak M2,2x6,5	6	DIN 7981			0,0286g
7	Matica M2	4	ISO 4032			0,02g
6	Vijak M2x12	4	ISO 4762			0,056g
5	Kamera	1				28 g
4	Nosač kamere	1	SK-003-001	PLA		2 g
3	Nosač gornjeg motora	1	SK-002-000	PLA		2 g
2	Servo motor	2				10 g
1	Nosač donjeg motora	1	SK-001-000	PLA		9 g

Broj naziva - code

	Datum	Ime i prezime	Potpis
Projektirao	10.1.2017.	Juraj Benić	
Razradio	10.1.2017.	Juraj Benić	
Crtao	10.1.2017.	Juraj Benić	
Pregledao			

FSB Zagreb

ISO - tolerancije: Ø3 H7 (0,01 / 0)

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Materijal:

Masa: 62 g

Naziv: Sustav za pomicanje kamere

Pozicija: 1

Format: A2

Mjerilo originala: M1:1

Listova: 13

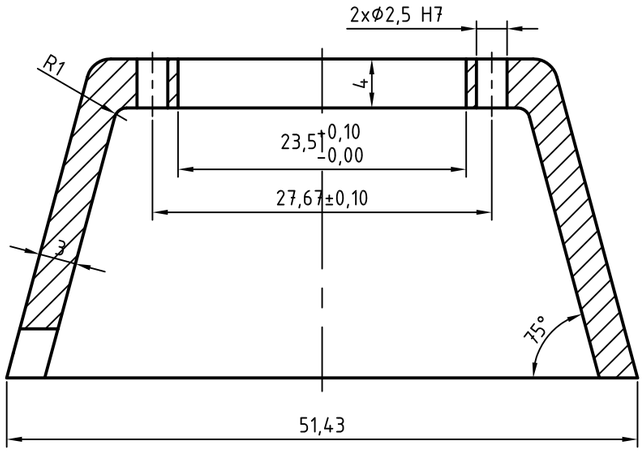
Crtež broj: SK-000-002

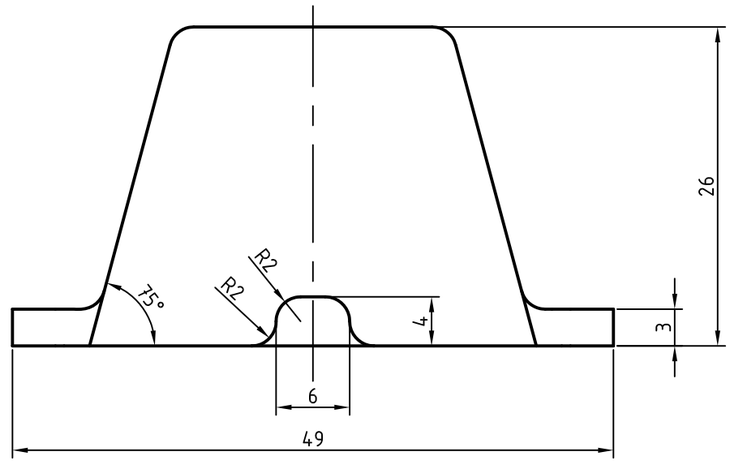
List: 2

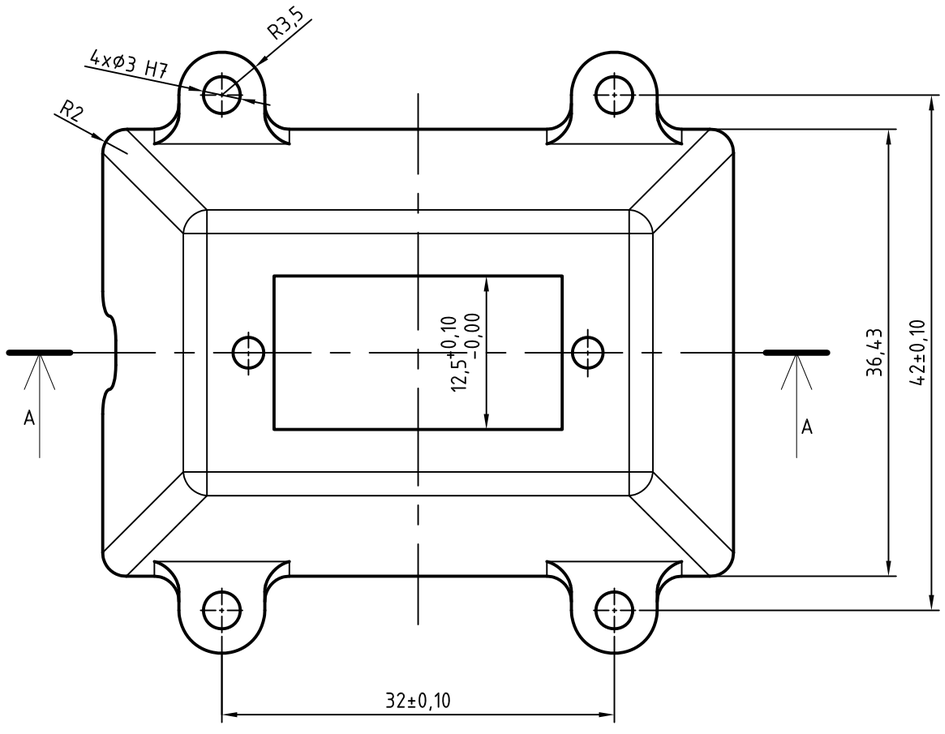
Design by CADLab

Ra 6,3

PRESJEK A-A







ISO - TOL	
Ø2,5 H7	0,01
	0
Ø3 H7	0,01
	0

	Datum	Ime i prezime	Potpis
Projektirao	26.12.2016.	Juraj Benić	
Razradio	26.12.2016.	Juraj Benić	
Crtao	26.12.2016.	Juraj Benić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena: Svi nekotirani radiusi R2

Kopija

Materijal: PLA

Masa: 9 g

Mjerilo originala M2:1

Naziv: Nosač donjeg motora

Pozicija: 1

Format: A3

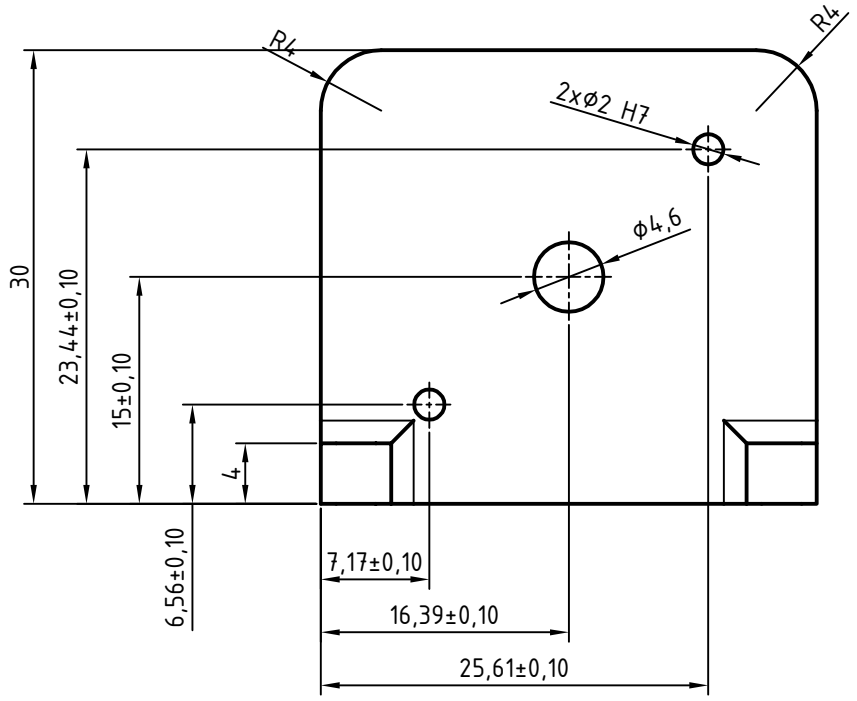
Listova: 13

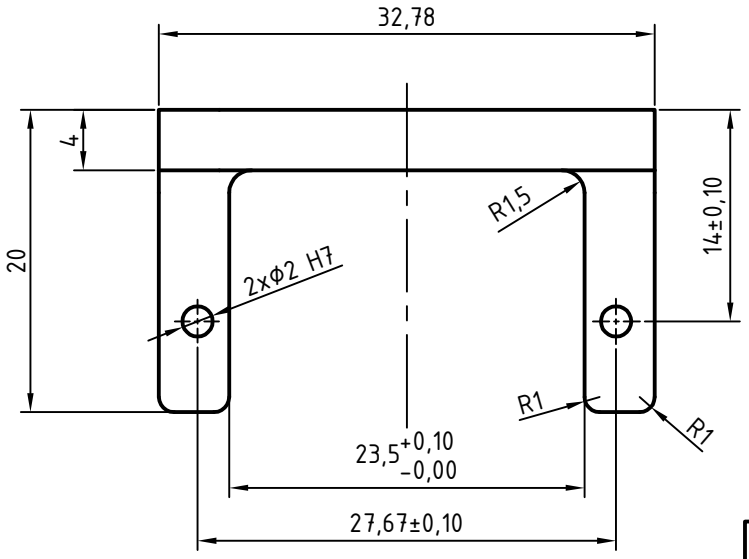
List: 3

Crtež broj: SK-001-000

0 10 20 30 40 50 60 70 80 90 100

Ra 6,3





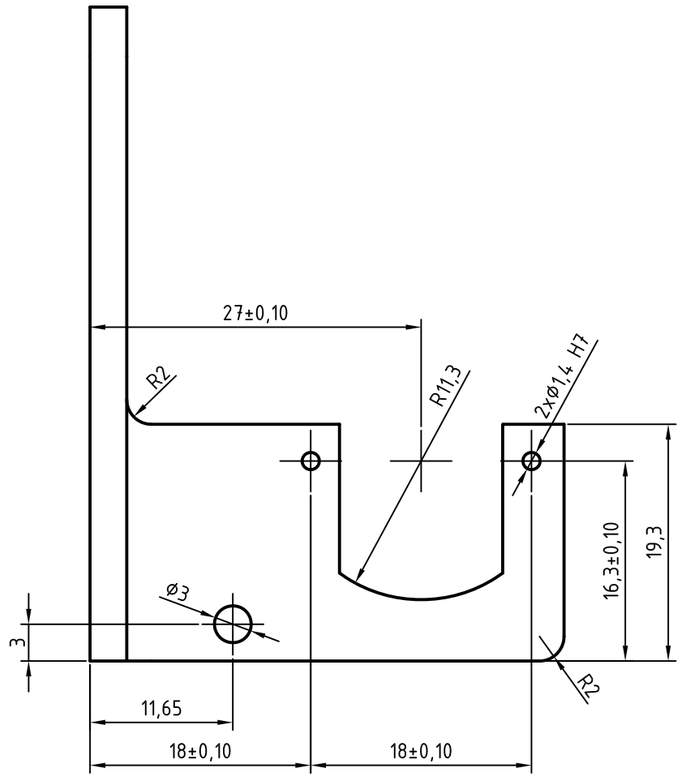
ISO - TOL	
ø2 H7	0,01
	0

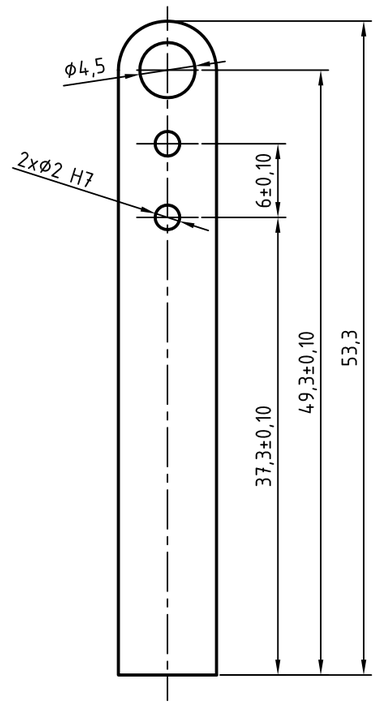
	Datum	Ime i prezime	Potpis
Projektirao	26.12.2016.	Juraj Benić	
Razradio	26.12.2016.	Juraj Benić	
Crtao	26.12.2016.	Juraj Benić	
Pregledao			

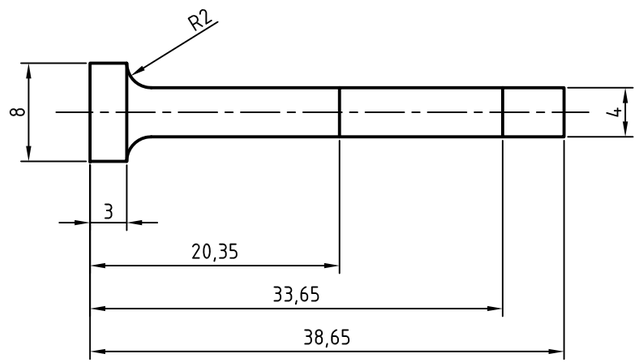
FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena: Svi nekotirani radiusi R1,5	Kopija
Materijal: PLA — Masa: 2 g	
Mjerilo originala: M2:1	Naziv: Nosač gornjeg motora — Pozicija: 3
Crtež broj: SK-002-000	Format: A4 — Listova: 13 — List: 4

Ra 6,3







ISO - TOL	
ø1,4 H7	0,01
	0
ø2 H7	0,01
	0

	Datum	Ime i prezime	Potpis
Projektirao	10.1.2017.	Juraj Benić	
Razradio	10.1.2017.	Juraj Benić	
Crtao	10.1.2017.	Juraj Benić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Materijal: PLA

Masa: 2 g

Mjerilo originala

M2:1

Naziv:

Nosač kamere

Pozicija: 4

Format: A3

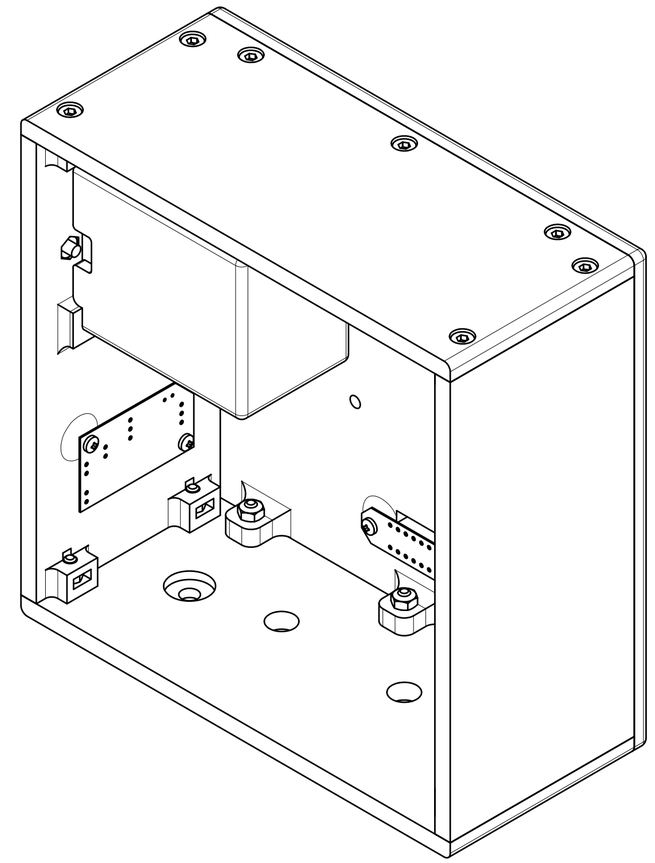
Listova: 13

List: 5

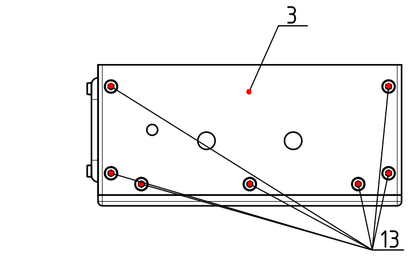
Crtež broj: SK-003-001

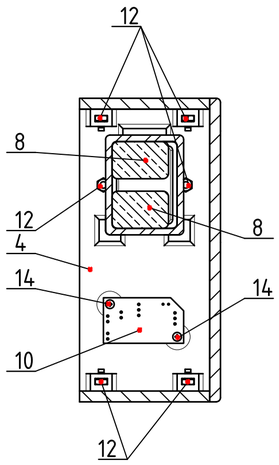
Design by CADLab

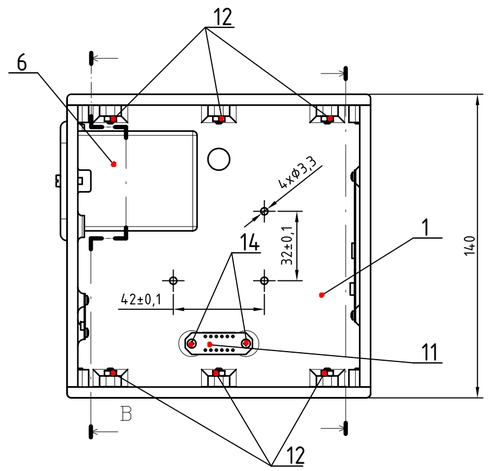
0 10 20 30 40 50 60 70 80 90 100

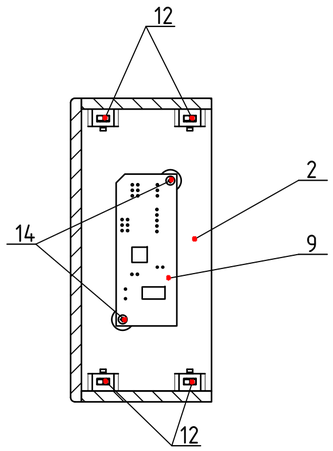


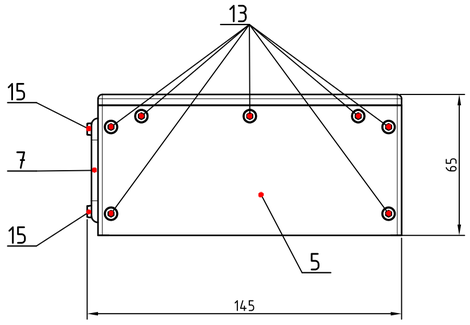
IZOMETRIJA
M1:1











Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
15	Vijam M3x10	2	ISO 1207			
14	Vijam st2 2x4 5 ch	6	ISO 7049			
13	Vijam M3x10	14	ISO 4762			
12	Matica M3	16	ISO 4032			
11	Kamera - bluetooth	1				
10	Kamera - napajanje	1	K-08-00			
9	Kamera - kontroler	1	K-07-00			
8	Baterija 9V	2				
7	Poklopac za baterije	1	KU-007-000	PLA		6 g
6	Kutija za baterije	1	KU-006-000	PLA		21 g
5	Zadnja strana kućišta	1	KU-005-000	PLA		19 g
4	Desna strana kućišta	1	KU-004-000	PLA		17 g
3	Prednja strana kućišta	1	KU-003-000	PLA		37 g
2	Lijeva strana kućišta	1	KU-002-000	PLA		26 g
1	Gornja strana kućišta	1	KU-001-000	PLA		61 g

Broj naziva - code

	Datum	Ime i prezime	Potpis
Projektirao	10.1.2017.	Juraj Benić	
Razradio	10.1.2017.	Juraj Benić	
Crtao	10.1.2017.	Juraj Benić	
Pregledao			

FSB Zagreb

ISO - tolerancije

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Materijal:

Masa: 218 g

Mjerilo originala

M1:2

Naziv:

Sklop kutije

Pozicija: 2

Format: A2

Listova: 13

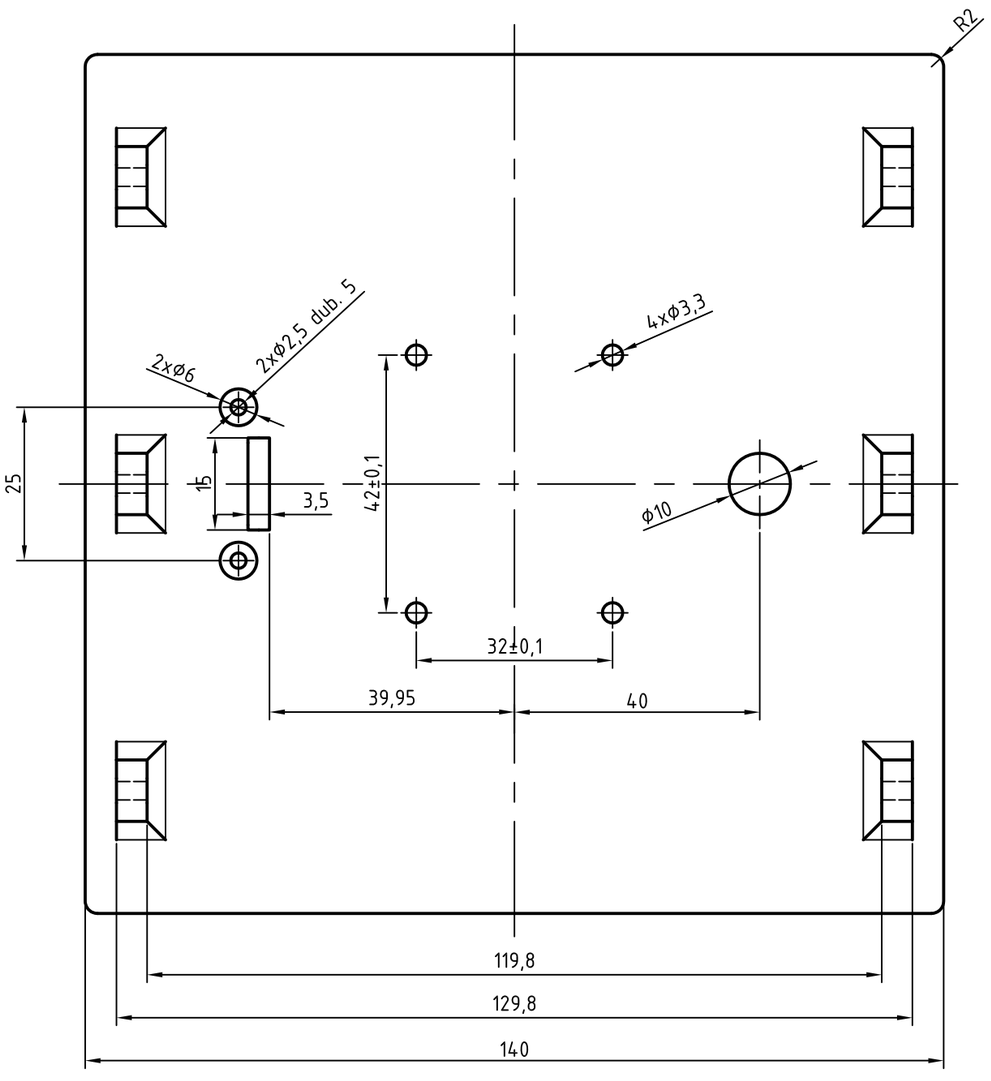
Crtež broj: KU-000-001

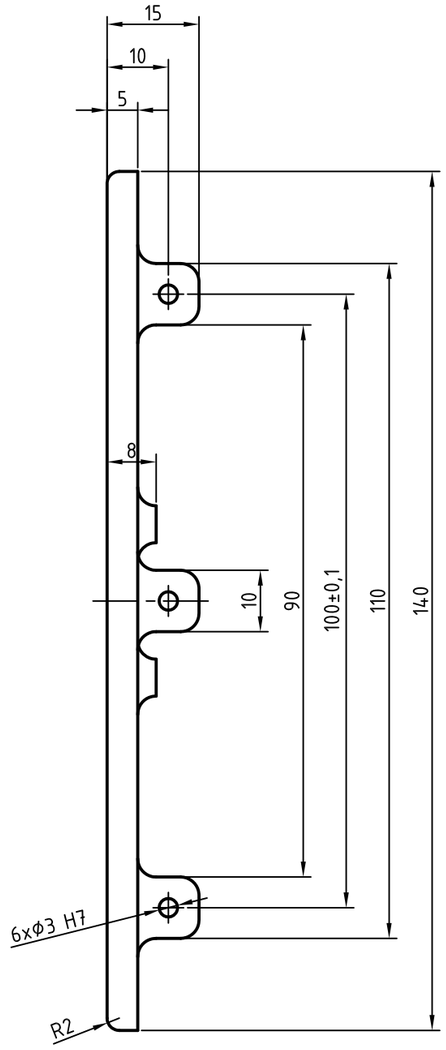
List: 6

0 10 20 30 40 50 60 70 80 90 100

Design by CADLab

Ra 6,3





ISO - TOL	
ø3 H7	0,01
	0

	Datum	Ime i prezime	Potpis
Projektirao	27.12.2016.	Juraj Benić	
Razradio	27.12.2016.	Juraj Benić	
Crtao	27.12.2016.	Juraj Benić	
Pregledao			

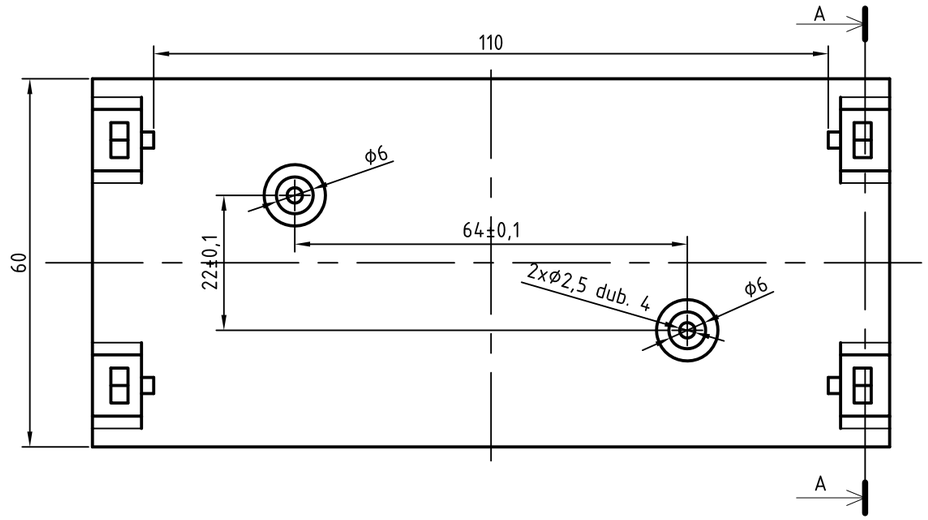
FSB Zagreb

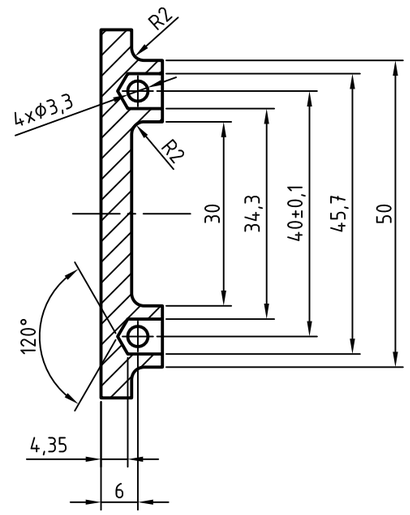
Objekt:		Objekt broj:	
		R. N. broj:	
Napomena: Svi nekotirani radiusi R3			Kopija
Materijal: PLA	Masa: 61 g		
Mjerilo originala M1:1	Naziv: Gornja strana kućišta	Pozicija: 1	Format: A3
			Listova: 13
	Crtež broj: KU-001-000		List: 7

Design by CADLab

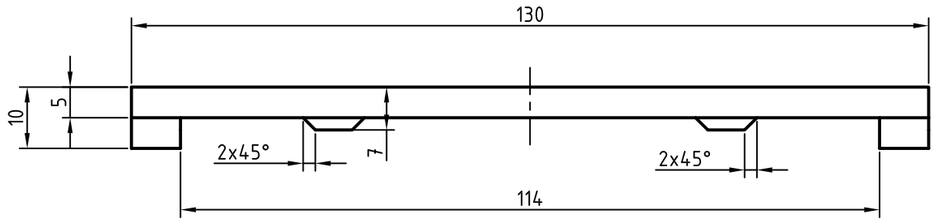
0 10 20 30 40 50 60 70 80 90 100

Ra 6,3





PRESJEK A-A



	Datum	Ime i prezime	Potpis
Projektirao	28.12.2016.	Juraj Benić	
Razradio	28.12.2016.	Juraj Benić	
Crtao	28.12.2016.	Juraj Benić	
Pregledao			

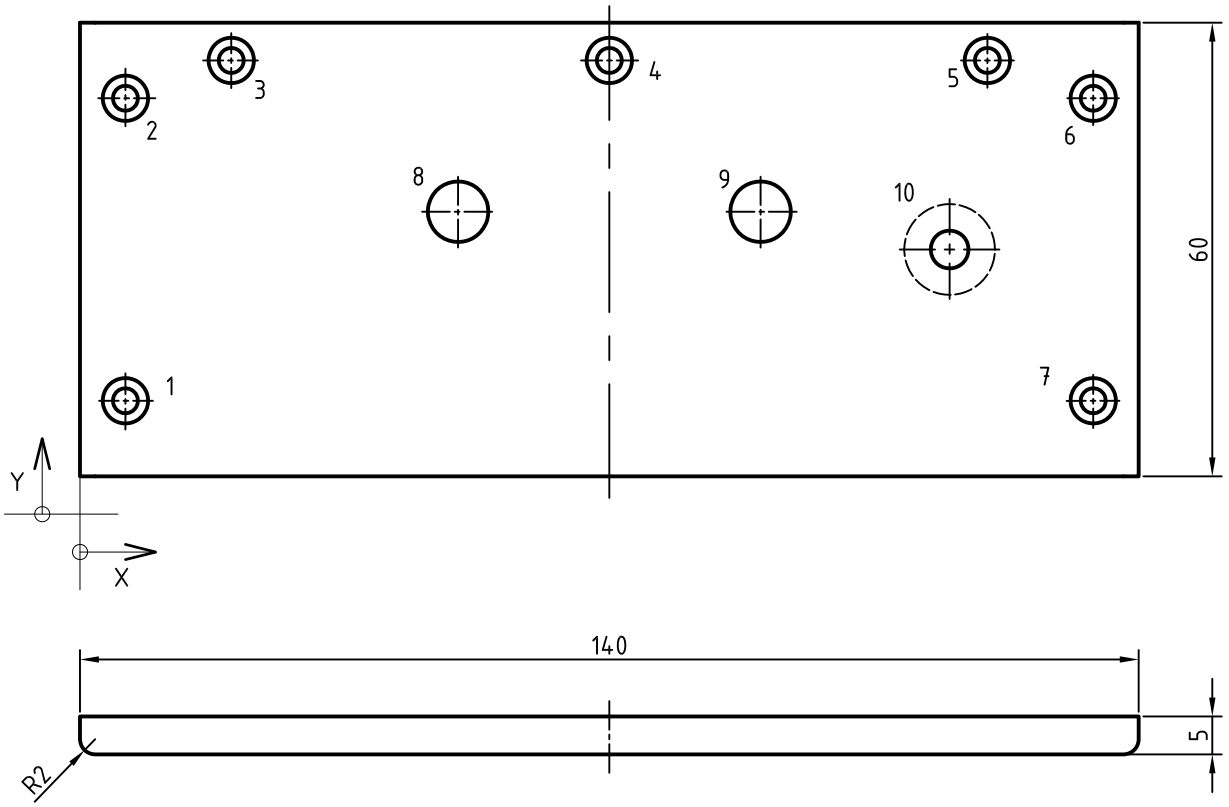
FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: PLA	Masa: 26 g
Mjerilo originala M1:1	Naziv: Lijeva strana kućišta
Pozicija: 2	Format: A3
Listova: 13	List: 8
Crtež broj: KU-002-000	

Design by CADLab

0 10 20 30 40 50 60 70 80 90 100

Ra 6,3



	X	Y	ϕ
1	6	15	ϕ3,3 upust za M3 ISO 4762
2	6	55	ϕ3,3 upust za M3 ISO 4762
3	20	60	ϕ3,3 upust za M3 ISO 4762
4	70	60	ϕ3,3 upust za M3 ISO 4762
5	120	60	ϕ3,3 upust za M3 ISO 4762
6	134	55	ϕ3,3 upust za M3 ISO 4762
7	134	15	ϕ3,3 upust za M3 ISO 4762
8	50	35	ϕ8 H7
9	90	35	ϕ8 H7
10	115	30	ϕ5 H7 upust odozdo ϕ12 H7 2,4 dub

ISO - TOL		
ϕ5 H7	0,012	0
ϕ8 H7	0,015	0
ϕ12 H7	0,018	0

	Datum	Ime i prezime	Potpis
Projektirao	27.12.2016.	Juraj Benić	
Razradio	27.12.2016.	Juraj Benić	
Crtao	27.12.2016.	Juraj Benić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena: Tolerancija pozicije provrta ±0,1

Kopija

Materijal: PLA

Masa: 37 g

Mjerilo originala

M1:1

Naziv:

Prednja strana kućišta

Pozicija: 3

Format: A4

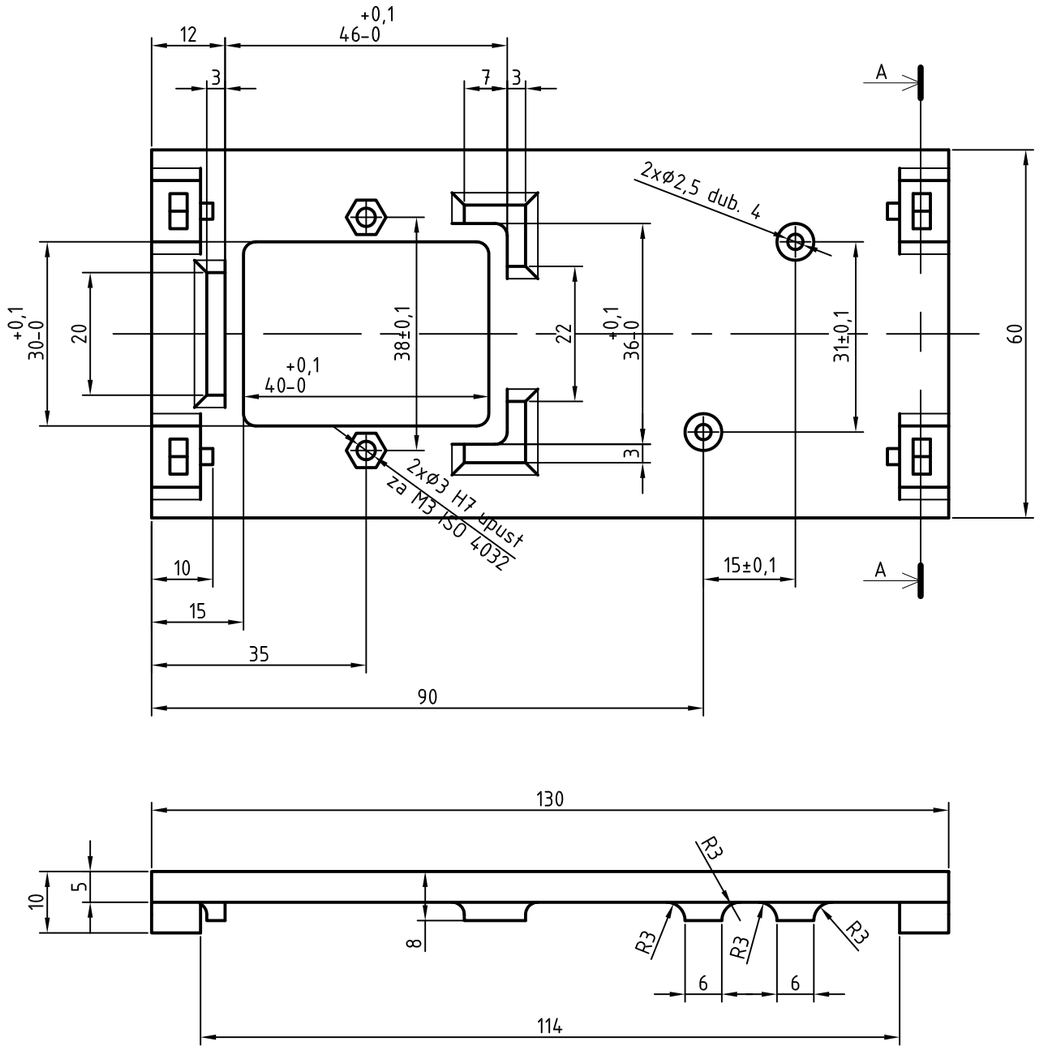
Listova: 13

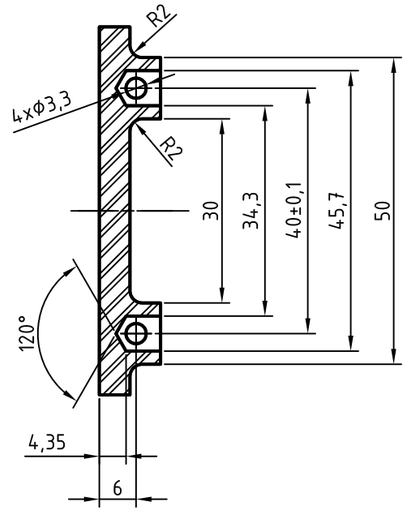
Crtež broj: KU-003-000

List: 9

Design by CADLab

Ra 6,3





ISO - TOL	
ø3 H7	0,01 / 0

	Datum	Ime i prezime	Potpis
Projektirao	28.12.2016.	Juraj Benić	
Razradio	28.12.2016.	Juraj Benić	
Crtao	28.12.2016.	Juraj Benić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena: Svi nekotirani radiusi R2

Kopija

Materijal: PLA

Masa: 17 g

Naziv: Desna strana kućišta

Pozicija: 4

Format: A3

Mjerilo originala: M1:1

Listova: 13

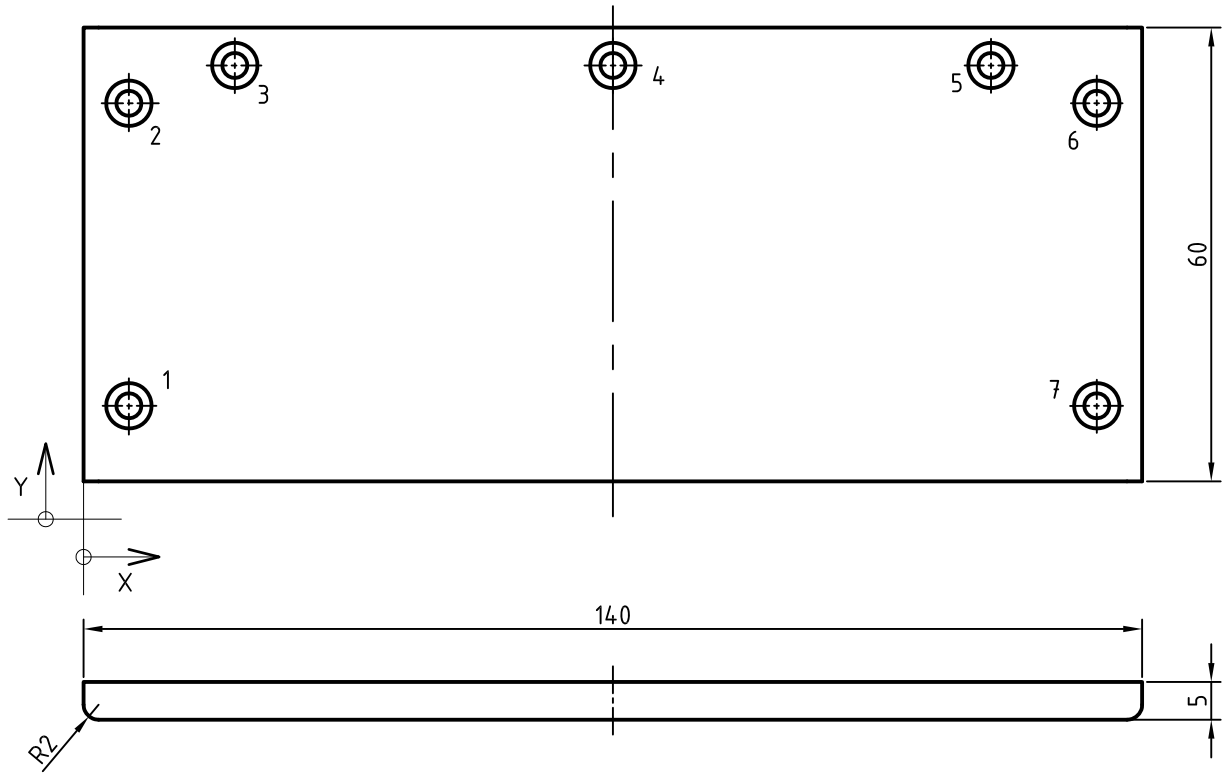
Crtež broj: KU-004-000

List: 10

Design by CADLab

0 10 20 30 40 50 60 70 80 90 100

Ra 6,3



	X	Y	⌀
1	6	15	⌀3,3 upust za M3 ISO 4762
2	6	55	⌀3,3 upust za M3 ISO 4762
3	20	60	⌀3,3 upust za M3 ISO 4762
4	70	60	⌀3,3 upust za M3 ISO 4762
5	120	60	⌀3,3 upust za M3 ISO 4762
6	134	55	⌀3,3 upust za M3 ISO 4762
7	134	15	⌀3,3 upust za M3 ISO 4762

	Datum	Ime i prezime	Potpis
Projektirao	27.12.2016.	Juraj Benić	
Razradio	27.12.2016.	Juraj Benić	
Crtao	27.12.2016.	Juraj Benić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena: Tolerancija pozicije provrta ±0,1

Kopija

Materijal: PLA

Masa: 19 g

Mjerilo originala

M1:1

Naziv: Zadnja strana kućišta

Pozicija: 5

Format: A4

Listova: 13

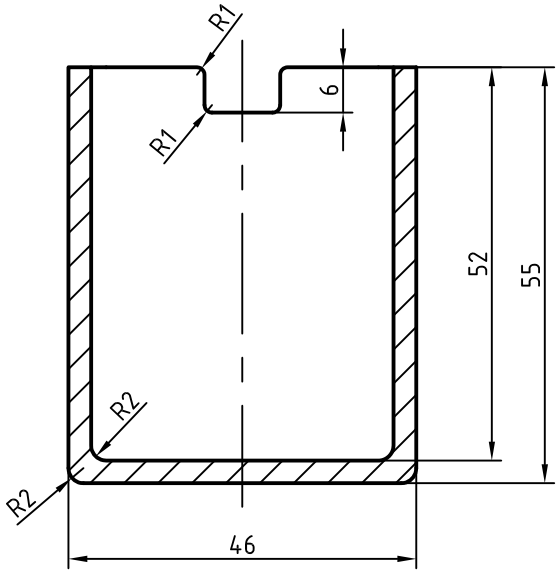
Crtež broj: KU-005-000

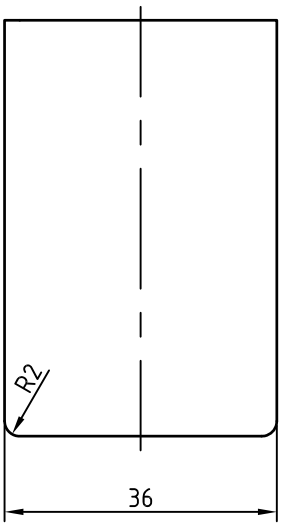
List: 11

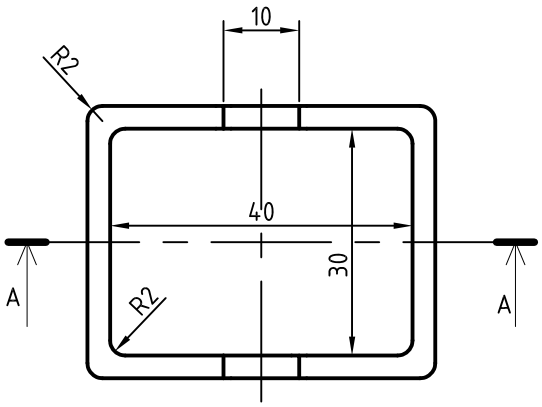
Design by CADLab

Ra 6,3

PRESJEK A-A





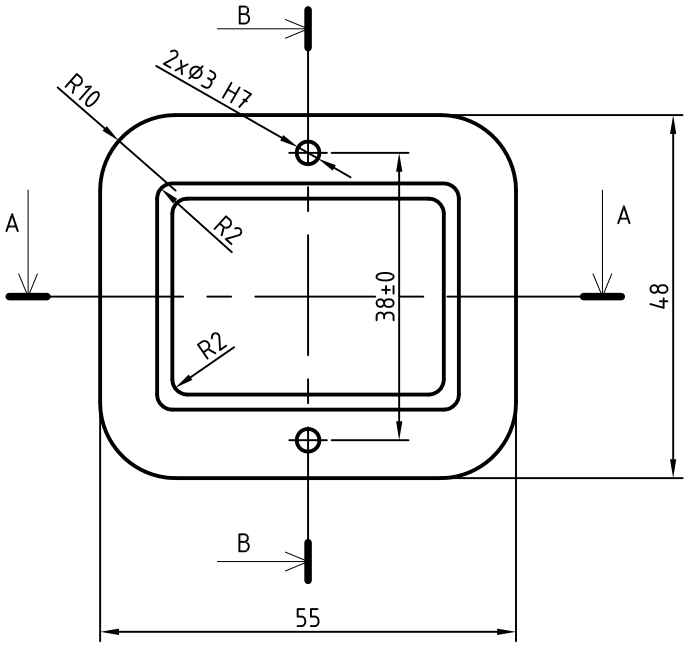


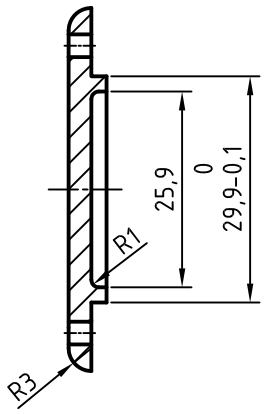
	Datum	Ime i prezime	Potpis
Projektirao	27.12.2016.	Juraj Benić	
Razradio	27.12.2016.	Juraj Benić	
Crtao	27.12.2016.	Juraj Benić	
Pregledao			

FSB Zagreb

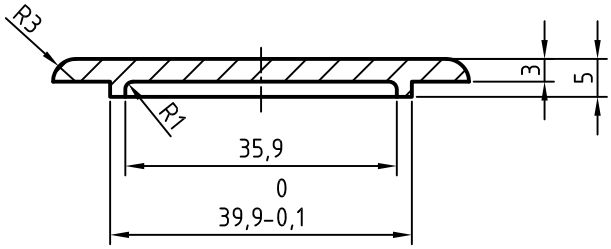
Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: PLA	Masa: 21 g
Mjerilo originala M1:1	Naziv: Kutija za baterije
Pozicija: 6	Format: A4
Listova: 13	List: 12
Crtež broj: KU-006-000	

Ra 6,3





PRESJEK B-B



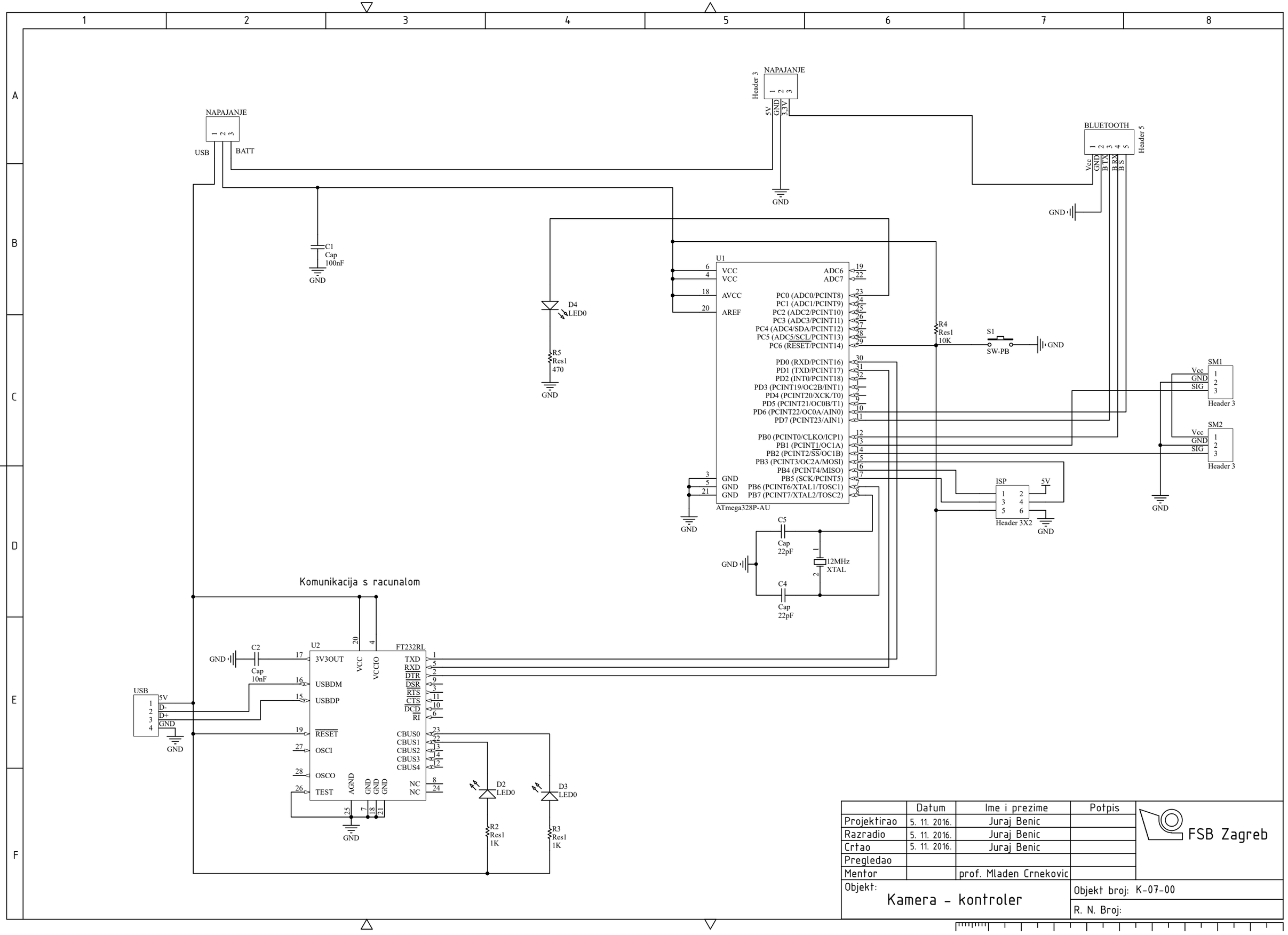
PRESJEK A-A

ISO - TOL	
ø3 H7	0,01
	0

	Datum	Ime i prezime	Potpis
Projektirao	27.12.2016.	Juraj Benić	
Razradio	27.12.2016.	Juraj Benić	
Crtao	27.12.2016.	Juraj Benić	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: PLA	Masa: 6 g
Mjerilo originala M1:1	Naziv: Poklopac za baterije
Pozicija: 7	Format: A4
Listova: 13	List: 13
Crtež broj: KU-007-000	



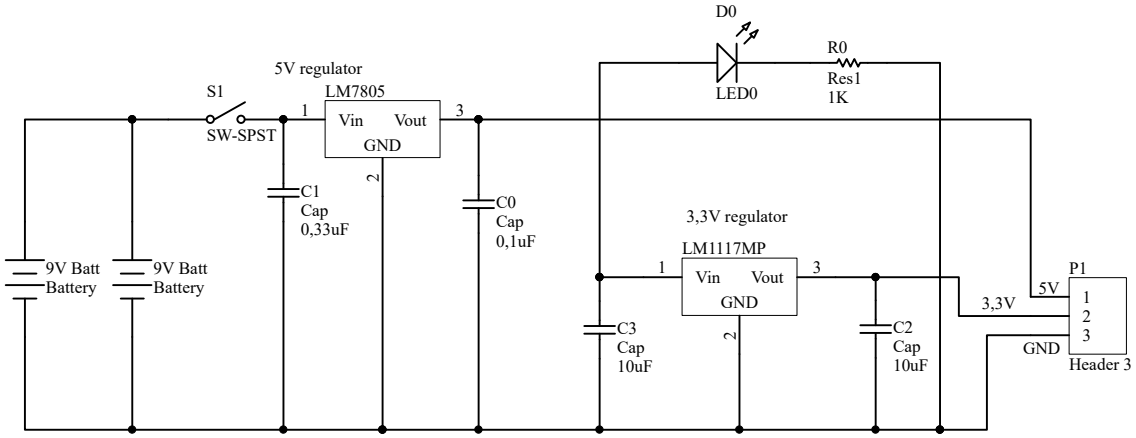
	Datum	Ime i prezime	Potpis
Projektirao	5. 11. 2016.	Juraj Benic	
Razradio	5. 11. 2016.	Juraj Benic	
Crtao	5. 11. 2016.	Juraj Benic	
Pregledao			
Mentor		prof. Mladen Crnekovic	

FSB Zagreb

Objekt: Kamera - kontroler

Objekt broj: K-07-00

R. N. Broj:



	Datum	Ime i prezime	Potpis
Projektirao	5. 11. 2016.	Juraj Benic	
Razradio	5. 11. 2016.	Juraj Benic	
Crtao	5. 11. 2016.	Juraj Benic	
Pregledao			
Mentor		prof. Mladen Crnekovic	

FSB Zagreb

Objekt:

Kamera - napajanje

Objekt broj: K-08-00

R. N. Broj: